ALGORITHMS FOR LOCATING FACILITIES UNDER UNCERTAINTIES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Chandrashekhar Nagarajan

January 2009

ALGORITHMS FOR LOCATING FACILITIES UNDER UNCERTAINTIES

Chandrashekhar Nagarajan, Ph.D.

Cornell University 2009

One of the main challenges in the area of discrete optimization is to find efficient and effective ways of solving problems that arise in day-to-day life. Traditionally, algorithms for such problems require complete knowledge of input parameters which is often undesirable and unrealistic. In this dissertation we consider some well-known *hard* location problems when the input parameters are not completely known in advance and design efficient algorithms for such scenarios guaranteeing quality of output solutions.

In the first part of the dissertation we give a general framework and algorithmic approach for incremental approximation algorithms. Given a notion of ordering on solutions of different cardinalities, we give solutions for all cardinalities such that the solutions respect the ordering and our solution is close in value to the value of an optimal solution of cardinality k for all values of k. We apply our framework to the incremental version of the k-median problem, k-MST problem, k-vertex cover problem, k-set cover problem and the facility location problem and give new or improved incremental algorithms for these problems. We also show that our framework applies to hierarchical clustering problems.

In the second part we consider the problem of leasing facilities over time where a newly arriving demand has to be either assigned to a previously leased open facility or to a newly leased facility. The serving cost of a demand can be defined as its distance from its assigned facility. The goal of the problem is to buy a set of leases at different facilities that minimizes the sum of leasing and serving costs. We give the first constant factor approximation algorithm for the offline version of the problem achieving a factor of 3. We also give the first deterministic algorithm for the online version that is $O(K \log n)$-competitive where K is the number of available facility leases and n is the number of clients.

We also compare the running times and quality of the solutions given by our incremental and hierarchical k-median algorithms with existing algorithms on different k-median datasets and verify that the quality of our solutions are better than the solutions of existing algorithms.

BIOGRAPHICAL SKETCH

Chandrashekhar Nagarajan was born on April 22, 1982 in Coimbatore, India. After finishing his schooling from Sri Ramakrishna Mission Higher Secondary School (South), Chennai in July 1999, he joined the Computer Science and Engineering Department for his undergraduate studies at Indian Institute of Technology, Madras (IITM) in Chennai where he was introduced to world of algorithms and optimization. He received his B.Tech (Bachelor of Technology) degree in Computer Science and Engineering from IITM in May 2003.

He then moved to Ithaca, NY in USA to pursue his doctorate degree in the field of Operations Research at the School of Operations Research and Information Engineering, Cornell University in the area of optimization. He was awarded a special Masters degree (Master of Science) by the School of Operations Research and Information Engineering, Cornell University in February 2006.

Upon completion of his Ph.D he will join the Search and Advertising Sciences Group of Yahoo Inc. at Santa Clara, CA as a Scientist.

To Amma, Appa and Meenu.

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Professor David P. Williamson for his generous support and invaluable guidance throughout the entire duration of my Ph.D. He has been an excellent advisor, friend, collaborator and teacher during the past five years. I would also like to thank Professor David P. Williamson for financially supporting my research through NSF grant CCF-0514628. I am also thankful to the School of Operations Research and Information Engineering, Cornell University for the generous teaching assistantships I received during my Ph.D.

I would like to specially thank my committee members Professor Evà Tardos and Professor Huseyin Topaloglu for their advice on my research and career and for their support throughout my Ph.D. I wish to thank Professor David Shmoys for his suggestions and comments regarding different research topics during different times of my Ph.D. I would also like to thank Professor Robert Bland for his letters of recommendation during my job application. I am extremely thankful to Eric Johnson for all his help in providing me with the computation resources for my computational experiments.

My stay in Ithaca would not have been as exciting and memorable as it was if not for the numerous friends around me. I would like to first thank Vidhyashankar and Karthick for their wonderful company during my entire stay here in Ithaca. I would also like to thank all my friends - Sethu, Venkat, Janani, Pavithra, Prajna, Mahesh, Surabhi, Aruna, Sowmya, Sharadha, Ganesh, Niranjan, Ishani, Bhargavi, Kiddo, Giri and Lakshmi who made my graduate life such a great, unforgettable experience. I would also like to thank Bikram, Souvik, Romita, Pathanil, Peter, Frans, Anke and Tim for making my stay at the department such a pleasant experience.

Finally and most importantly, words cannot express how grateful I am to have such a wonderful, caring and affectionate Appa, Amma and sister who have always been on my side all along the course of my Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

One of the main challenges in the area in discrete optimization is to find efficient and effective ways of solving problems that arise frequently in day-to-day life. Traditionally, algorithms for such problems require complete knowledge of the input parameters. However this assumption of complete knowledge in many practical scenarios is often undesirable and unrealistic. In this dissertation we consider some well-known *hard* optimization problems when the knowledge of input parameters is not completely known in advance and design efficient algorithms for such scenarios guaranteeing quality of the output solution.

1.1 Location Problems

In this dissertation we focus on location problems and consider uncertainties in the input parameters for such problems. Location problems occur in day-to-day life and model design situations such as efficient placement of factories, warehouses, schools, restaurants and hospitals and some modern day applications such as efficient placement of web servers. Here we focus on two very related problems in location theory: *the k-median problem* and *the uncapacitated facility location problem*.

Given a set of clients and a set of potential facilities in a metric space[1], the k-median problem seeks to find a set of k of these facilities to open so as to minimize the sum of the distances of the clients to the nearest open facility. In the uncapacitated facility location problem, we also have a cost for opening each

[1]The distances are symmetric and satisfy triangle inequality.

of the facilities and the goal is to find a set of facilities to open so as to minimize the sum of costs of opening these facilities and sum of the distances from each client to the nearest open facility with no explicit restriction on the number of facilities.

The k-median problem and the related uncapacitated facility location problem have been the objects of intense study in the algorithms community in the past few years. Both these problems belong to a class of difficult problems called *NP-hard* problems. There are no known algorithms that run in time polynomial in the size of input and output the optimal solution for any NP-hard problem. So we turn our attention to algorithms that run in polynomial time and find solutions that are *close* to the optimal solution. An *α-approximation algorithm* for a minimization problem runs in polynomial time and finds a solution whose objective value is at most α times the objective value of the optimal solution. This factor α is called the approximation ratio or the performance guarantee of the approximation algorithm. All optimization problems considered in this dissertation are NP-hard minimization problems.

The first approximation algorithm for the uncapacitated facility location problem was a greedy algorithm achieving a guarantee of $O(\log n)$ in the approximation ratio given by Hochbaum [38]. Hochbaum's algorithm works even when the distances do not obey triangle inequality. The first constant factor approximation algorithm for this problem is given by Shmoys, Tardos and Aardal [57] achieving a ratio of 3.16 and is later improved to $(1 + 2/e)$ by Chudak and Shmoys [23]. However these algorithms are based on LP-rounding where we have to solve the standard linear program of the uncapacitated facility location problem optimally and round the fractional solution to an integer solution.

Since we need to solve the linear program explicitly, such algorithms have high running times.

Jain and Vazirani give a *primal-dual* algorithm achieving an approximation factor of 3. A primal-dual algorithm maintains a feasible primal integral solution and a feasible dual solution such that the cost of the primal solution is at most some factor α times the cost of the dual solution. Since the dual solution cost is a lower bound on the cost of the optimal solution the primal integral solution achieves an approximation factor of α. Jain and Vazirani also observe that the uncapacitated facility location problem can be viewed as a Lagrangean relaxation of the k-median median problem and utilize this to give a 6-approximation algorithm for the k-median problem. An algorithm for the facility location problem which can be modified to exploit the Lagrangean relaxation to solve the k-median problem in this way is called a *Lagrangean multiplier preserving (LMP)* facility location algorithm (see 2.5.1 for more details).

Jain, Mahdian, Markakis, Saberi, and Vazirani [41] give two greedy uncapacitated facility location algorithms with performance guarantees of 1.861 and 1.61 and analyze them using *dual-fitting*. A dual-fitting algorithm maintains an infeasible dual solution and a feasible primal solution such that the cost of primal solution is at most the cost of the dual solution. Then the dual solution is made feasible by scaling it down by a factor α. Since the cost of any feasible dual solution is at least the cost of the optimal solution, the primal solution is an α approximate solution. Jain et al. also give a LMP 2-approximation algorithm for the problem which gives a 4-approximation algorithm for the k-median problem. The approximation ratios for the uncapacitated facility location problem were improved to 1.51 by Mahdian, Ye, and Zhang [46] and then to 1.5 by

Byrka [16]. Regarding the hardness of the problem, Guha and Khuller [34] showed that the best approximation factor possible for this problem is 1.463, assuming $NP \nsubseteq DTIME[n^{O(\log\log n)}]$.

Lin and Vitter [45] first consider the metric k-median problem and give an algorithm that, for any $\epsilon > 0$ finds a solution of cost no more than $2(1+\epsilon)$ times the optimum, while opening at most $(1+1/\epsilon)k$ facilities. The first approximation algorithm that produces a feasible solution is a randomized algorithm of $O(\log n \log\log n)$ due to Bartal [7] by approximating any metric to a tree metric. This algorithm was later derandomized and refined to an approximation ratio of $O(\log k \log\log k)$ by Charikar, Chekuri, Goel, and Guha [18]. The first constant factor approximation algorithm of $6\frac{2}{3}$ was obtained by Charikar, Guha, Tardos, and Shmoys [19] using a LP-rounding technique.

The LMP facility location algorithms of Jain and Vazirani [42] and Jain et al. [41] improve the approximation factor of the k-median problem to 6 and 4 respectively. The currently best known approximation algorithm for the k-median problem is given by Arya, Garg, Khandekar, Meyerson, Munagala and Pandit [6]. This is a local-search based algorithm in which one starts with some feasible solution and repeatedly perform local changes to improve the cost until no such improvements exist. Arya et al. prove that this algorithm runs in polynomial time and has an approximation ratio of $3+\epsilon$.

1.2 Modelling Uncertainty

The classical approach to any optimization problem is to model it by assuming that the input data is known completely and precisely. However, for most problems getting accurate input data is often very expensive and in many cases impossible. This led to extensive research in the past decade to develop new models for various problems to capture uncertainties in the input parameters and design algorithms for them.

One way to model this uncertainty is to formulate the problem as a stochastic program when the probability distribution of inputs are known or can be estimated. Stochastic programming refers to a general class of optimization problems when the inputs have uncertainty and are modelled by a probability distribution on inputs. The algorithm outputs a solution before the realization of the inputs and the cost of the solution is computed depending on the revealed input scenario. The objective here is to minimize the expected cost of the decision taken over the distribution of the input scenarios. Karger and Minkoff [44] consider the Steiner tree problem with each node wishing to get connected to the tree independently with some fixed probability. They give an algorithm which outputs a set of paths connecting each client to the root and this path that will be used if the client becomes active. They also prove that the expected total weight of used edges is at most a constant factor away from the best network which connects the active clients to the root.

Two-stage stochastic optimization with recourse is a natural extension to the stochastic programming model for capturing uncertainty in the input. Here again the input parameters are uncertain and are given by a known distribution on the scenarios. However, the decision is made in two stages – once before the the input scenario is revealed and then again as a recourse after revelation. The second stage decisions are typically costlier than the first stage decisions since they typically involve rapid reaction to the revealed scenario. The objective is to minimize the sum of the first stage decision costs and the expected cost of the recourse decision taken over the distribution of the scenarios. Ravi and Sinha [56] and Immorlica, Karger, Minkoff and Mirrokni [40] consider the two stage stochastic versions of several classical problems such as facility location, shortest path and bin packing problems and give polynomial time approximation algorithms for them. Gupta, Pal, Ravi and Sinha [35] consider various two-stage problems with recourse for arbitrary distribution using black-box model[2] with the restriction that the second-stage costs be proportional to the first-stage costs. The two-stage model can be naturally generalized to multi-stage by adding additional recourse stages each consisting of an observation and a decision responding to it. Swamy and Shmoys [61] give approximation algorithms for many of the multi-stage stochastic integer programs using a natural LP-rounding approach.

On the other hand, *robust optimization* deals with worst case optimization in the presence of uncertain and inaccurate input data. In a robust optimization problem, bounds on various input parameters are given and the goal is to find a solution that remains feasible and minimizes the cost in the worst-case scenario. Hence, robust optimization can be considered as the worst-case analogue of stochastic optimization. Ben-Tal and Nemrovski [9, 10] consider robust linear programming and convex optimization problems and model the uncertainty in data by assuming that the data is drawn from ellipsoids of data points and proposed efficient algorithms in such scenarios. Bertsimas and Sim [12] consider uncertainty in cost functions for optimization problems and give algorithms for the robust counterpart of any discrete optimization problem. Nikulin's annotated bibliography [51] gives a detailed list of references in the area of combi-

[2]any number of independent input samples can be drawn from the distribution

natorial optimization and scheduling theory concerning robustness and other techniques dealing with worst case optimization under uncertainty of input parameters.

The study of *online algorithms* considers problems in which the uncertainty in the input is modeled by revealing the input over time, and we have to make decisions when the input is revealed without knowledge of future inputs. So an online algorithm is measured against an adversarial sequence of input revelation. The *competitive ratio* of an online algorithm is the worst-case ratio of the objective function of the algorithm's solution to the offline optimal objective value over all possible inputs. The paging problem and the k-server problem are some of the earlier online problems considered by researchers a few decades ago (see Borodin and El-Yaniv [14]). Recently there has been a lot of research in studying and designing competitive algorithms for online versions of well-known optimization problems: Alon, Awerbuch, Azar, Buchbinder and Naor [1] (online set cover problem), Meyerson [49] (online facility location problem), Meyerson [50] (parking permit problem), Berman and Coulston (online Steiner tree problem [11]) to name a few.

Uncertainty in the cardinality of a cardinality constrained problem (like the k-median problem) is tackled slightly differently compared to other kinds of uncertainties. There is a special way to model the uncertainty in cardinality when the cardinality is always increasing as the time progresses. It is natural for a problem with this kind of uncertainty to build solutions in an incremental fashion. Here the complete knowledge of the inputs is known in advance except for the cardinality of the output solution. Such a cardinality constrained problem with uncertain, but increasing cardinality is called an *incremental problem*. An algorithm for an incremental problem outputs an ordering of solutions according to their cardinalities where lower cardinality solutions are constrained to be part of the higher cardinality solutions. An incremental algorithm is said to be α-competitive if the maximum ratio of the solution's cost to the optimal cost for all the cardinality is no more than α. Let us consider an example of the incremental k-median problem. A restaurant chain which wants to open some restaurants to serve its clients may not know immediately how many restaurants it wants to build eventually. However, the number of restaurants can only increase over time. In reality it may want to build a few restaurants at the start and later build additional restaurants as funds become available. This incremental k-median problem is first introduced by Mettu and Plaxton [48], who give a constant competitive algorithm for it. Incremental versions of various other problems are also considered and competitive algorithms have been designed over the past couple of decades; Gonzalez [33] (incremental k-center problem), Plaxton [55] (incremental facility location problem), Hartline and Sharp [36] (incremental flow problem) to name a few.

Jia, Lin, Noubir, Rajaraman and Sundaram [43] introduce yet another framework for dealing with uncertainty in the input called *universality* which guarantees goodness in the output for all possible input scenarios. The universal version of Traveling Salesman Problem (TSP) is first considered by Platzman and Bartholdi [53]. A universal approximate solution for the traveling salesman problem is a tour over all the vertices such that for any subset of the vertices, the sub-tour induced by the subset approximated the optimal solution on those subset of vertices. Platzman and Bartholdi give an algorithm which gives a logarithmic approximation guarantee for the universal TSP in the Euclidian metric space. Jia et al. consider universal algorithms for metric TSP, Steiner tree and set cover problems and give universal algorithms with poly-logarithmic approximation guarantees for these problems.

In this dissertation we mainly consider input uncertainties in two very related location problems: the k-median problem and the uncapacitated facility location problem. We consider incremental versions of k-median problem and give algorithms with improved competitive ratio over the existing algorithm of Mettu and Plaxton [48]. Our algorithm can use any good k-median algorithm as a black box and cleverly combines the k-median solutions to generate a good incremental k-median solution. Then we extend the framework to incorporate incremental versions of different cardinality constrained minimization problems like the set cover, vertex cover, k-means, and facility location problems. We give a generic algorithm which gives competitive algorithms for each of the problems considered with performance guarantees better than existing algorithms. We also show that the framework and the generic algorithm can be applied to the hierarchical clustering problems. In particular, we give an improved algorithm for a hierarchical version of the k-median problem introduced by Plaxton [55].

We then consider the problem of leasing facilities over time, following the general infrastructure leasing problem framework introduced by Anthony and Gupta [3]. Here the clients arrive at different times seeking service for that particular time instant from a facility open during that time. If there are K different lease types available for the facilities, Anthony and Gupta give an $O(K)$-approximation algorithm for the problem. We are able to improve this to a 3-approximation algorithm by using a variant of the primal-dual facility location algorithm of Jain and Vazirani [42]. This algorithm is an offline algorithm where the arrivals of different clients are known in advance.

We also consider the online version of the facility leasing problem, in which the clients to be served arrive over time and are not known in advance. This problem generalizes both the online facility location problem (introduced by Meyerson [49]) and the parking permit problem (also introduced by Meyerson [50]). We give a deterministic algorithm for the problem that is $O(K \log n)$-competitive. No previous result was known for this problem. To achieve our result, we modify an $O(\log n)$-competitive algorithm of Fotakis [28] for the online facility location problem. We also reanalyze his algorithm via the dual-fitting technique to prove that it achieves the $O(\log n)$ competitive ratio.

We test our incremental and hierarchical k-median algorithms on different k-median datasets against the incremental k-median algorithm of Mettu and Plaxton [48] and the hierarchical k-median algorithm by Plaxton [54]. We use the k-median solutions from the LP rounding algorithm of Charikar et al. [19], the local search algorithm of Arya et al. [6] and the primal dual algorithm of Jain et al. [41] as inputs to our incremental and hierarchical k-median algorithm.

Chapter 2 gives the generic framework and the generic competitive algorithms for incremental problems for cardinality constrained problems and extends them to incorporate hierarchical constraints as well. This is joint work with Guolong Lin, Rajmohan Rajaraman and David P. Williamson. Chapter 3 gives offline and online algorithms for facility leasing problems and prove their performance guarantees. This is joint work with David P. Williamson. In chapter 4, we compare our incremental and hierarchical k-median algorithms with already existing algorithms by simulating them on different datasets. We give the experimental details and results of the simulations done on the datasets and compare the running times and quality of solutions of different algorithms.

CHAPTER 2

INCREMENTAL PROBLEMS

2.1 Introduction

2.1.1 Incremental problems

A company is building facilities in order to supply its customers. Because of limited capital, it can only build a few at this time, but intends to expand in the future in order to improve its customer service. Its plan for expansion is a sequence of facilities that it will build in order as it has funds. Can it plan its future expansion in such a way that if it opens the first k facilities in its sequence, this solution is close in value to that of an optimal solution that opens any choice of k facilities? The company's problem is the *incremental* k-median problem, and was originally proposed by Mettu and Plaxton [48][1]. The standard k-median problem has been the object of intense study in the algorithms community in the past few years. Given the locations of a set of facilities and a set of clients in a metric space, a demand for each client, and a parameter k, the *k-median problem* asks to find a set of k facilities to *open* such that the sum of the demand-weighted distances of the clients to the nearest open facility is minimized. In the incremental k-median problem, we are given the input of the k-median problem without the parameter k and must produce a sequence of the facilities. For each k, consider the ratio of the cost of opening the first k facilities in the ordering to the cost of an optimal k-median solution. The goal of the problem is to find an ordering that minimizes the maximum of this ratio over all values of k. An

[1]Mettu and Plaxton call it the *online* median problem, but we would like to draw a distinction between incremental and online problems.

algorithm for the problem is said to be α-*competitive* if the maximum of the ratio over all k is no more than α. This value α is called the *competitive ratio* of the algorithm. We will also consider randomized algorithms for the incremental k-median problem. For a randomized algorithm, we consider the ratio of the expected cost of opening the first k facilities in the ordering to the cost of an optimal k-median solution. The algorithm is α-competitive if this ratio is at most α for all k.

In a similar manner, one can also define natural incremental versions of any cardinality constrained minimization problems, including the k-minimum spanning tree problem (k-MST), k-vertex cover, and k-set cover problems. In the standard weighted vertex cover problem, we are given an undirected graph with weights on the vertices and we wish to find a minimum-weight subset of vertices S such that every edge has at least one endpoint in S. In the k-vertex cover problem, we wish to find a minimum-weight set of vertices that covers at least k edges. In the incremental k-vertex cover problem, we wish to find a sequence of vertices, such that if we choose the smallest prefix of vertices in the sequence that covers at least k edges, this solution is close in value to that of the optimal k-vertex cover solution. An incremental version of the facility location problem, which is not cardinality-constrained, has also been defined [55].

Perhaps less obviously, many hierarchical clustering problems can also be cast as incremental problems. In hierarchical clustering, we give clusterings with k clusters for all values of k by starting with n clusters and repeatedly merging selected pairs of clusters until all points are in a single cluster. Given some objective function on a k-clustering, again we would like to ensure that for any k, the cost of our k-clustering obtained in this way is not too far away from the cost of an optimal k-clustering. The connection with incremental problems is this: for the incremental k-median problem, we insist that for any k, k', with $k < k'$, our solution with k facilities is *ordered* with respect to our solution on k' facilities; namely, the smaller solution is a subset of the larger. In hierarchical clustering, for any k, k', with $k < k'$, our k-clustering must also be ordered with respect to our k'-clustering; namely, the k'-clustering must be a refinement of the k-clustering. We can then consider various clustering criteria: minimize the maximum radius from a cluster center (k-center), minimize the sum of demand-weighted distances of points to their cluster center (k-median), or minimize the sum of demand-weighted distances-squared of points to their cluster center (k-means). From these we obtain hierarchical variants, which we say are α-competitive if for any k, the k-clustering produced by our algorithm is at most α times the cost of an optimal k-clustering under the given objective.

2.1.2 Our contribution

The central contribution of this chapter is to give a general approach for solving incremental optimization problems. We then apply this to the incremental versions of the k-median, k-means, k-MST, facility location, k-vertex cover, and k-set cover problems. Furthermore, we apply it to hierarchical clustering problems with the k-median and k-means objective functions. We state our approach in terms of *posets* on solutions to the problems, in which two solutions are comparable in the poset if they obey the ordering imposed by the incremental solution (e.g. if one of the k-median solutions is a subset of the other, or one of the k-vertex cover solutions is a subset of the other, or one of the k-clusterings is a refinement of the other). Each solution in the poset has a cost, as defined by the underlying optimization problem. In addition, we associate a benefit with each solution that models the constraint of the optimization problem (corresponding, for example, to the number of unopened facilities, or the number of edges covered). The goal of the incremental problem is to find a chain of solutions such that for any b, the least element in the chain (according to the partial order) that has benefit at least b has cost close to that of an optimal solution of benefit at least b.

To obtain a solution that is competitive for a given incremental problem in polynomial-time, our algorithm relies on an α-approximation algorithm for the underlying offline optimization problem. It also relies on an *augmentation* subroutine that, given two solutions of benefits b, b', $b < b'$, which are incomparable in the poset, finds another solution of benefit at least b' that is comparable in the poset to the solution of benefit b. If one can show that this solution has cost no more than a linear combination of the costs of the original two solutions, then one can obtain an $O(\alpha)$-competitive algorithm, where the constant in the big-O depends on the constants in the linear combination. The basic idea of the incremental algorithm is to build a chain of solutions of geometrically increasing cost by repeatedly applying the augmentation subroutine to the current solution in the chain and a solution generated by the α-approximation algorithm that has cost no more than the next bound in the geometrically increasing order. Similar ideas are implicit in the minimum latency approximation algorithm of Blum, Chalsani, Coppersmith, Pulleyblank, Raghavan, and Sudan [13], the incremental facility location algorithm of Plaxton [55] and the hierarchical k-center algorithm of Dasgupta and Long [26]. Choosing a random shift of the geometrically increasing sequence as in Goemans and Kleinberg [32] and Dasgupta and Long [26] gives improved randomized algorithms.

In some cases, we are able to improve the competitive ratio still further. In particular, if there exists a Lagrangean multiplier preserving ρ-approximation algorithm for the problem in which Lagrangean relaxation has been applied to the benefit constraint, we are able to give the same result as above in which this algorithm replaces the α-approximation algorithm. This yields improved competitive ratios in the cases where we have such algorithms with performance guarantees ρ better than the best known performance guarantee α for the problem with the benefit constraint. In particular, there is a Lagrangean multiplier preserving 2-approximation algorithm for the facility location problem (due to Jain, Mahdian, Markakis, Saberi, and Vazirani [41]), which we can use in place of a $(3 + \epsilon)$-approximation algorithm for the k-median problem (due to Arya, Garg, Khandekar, Meyerson, Munagala, and Pandit [6]), yielding improvements in the competitive ratios for the incremental k-median problem, hierarchical k-median problem, and hierarchical k-means problem.

We summarize our main results for incremental problems in Table 2.1. The first column gives the best previously known competitive ratio for a polynomial-time algorithm. The second and third state the competitive ratio for incremental solutions obtained using optimal algorithms for the benefit-constrained problems and are thus non-polynomial-time algorithms; they should be viewed as existential results. The fourth and fifth state the competitive ratio for our polynomial-time algorithms via an α-approximation algorithm. The results with * were independently obtained by Chrobak, Kenyon, Noga, and Young [21]. Table 2.2 gives the competitive ratio for our polynomial-time algorithms via a Lagrangean multiplier preserving ρ-approximation algorithm.

Table 2.1: Summary of results for the incremental problems

	Competitive ratio				
		Via optimal		Via approx	
Problem	Prev known	Det	Rand	Det	Rand
Incremental k-median	29.86 [48]	8^*	$2e^*$	$24 + \epsilon^*$	$6e + \epsilon^*$
Incremental k-MST	8 [13], $2e$ [32]	4	e	8	$2e$
Incremental k-vertex cover		4	e	8	$2e$
Incremental k-set cover		4	e	$O(\log n)$	$O(\log n)$
Incremental facility location	$6 + \epsilon$ [55]	4	e	6	$1.5e$
Hierarchical k-median	238.88 [55]	20.71	10.03	$62.13 + \epsilon$	$30.09 + \epsilon$

Table 2.2: Summary of results for the incremental and hierarchical k-median problems

	Competitive ratio		
		Via LMP approx	
Problem	Prev known	Det	Rand
Incremental k-median	29.86 [48]	16	$4e$
Hierarchical k-median	238.88 [55]	41.42	20.06

2.1.3 Related work

All of the optimization problems studied in this chapter are NP-hard and have been extensively studied with respect to their approximability. Several approximation algorithms are known for the vertex cover and set cover problems (see, for example, [39, 62]). Our incremental k-vertex cover algorithm relies on a 2-approximation algorithm for k-vertex cover, while our incremental k-set cover algorithm relies on an $O(\log n)$-approximation algorithm for k-set cover [15, 30, 37, 47, 58].

The k-median problem and the related uncapacitated facility location problems have been the objects of intense study in the algorithms community in the past few years. As discussed in the introduction, the currently best known approximation algorithms for these problems have performance guarantees of $3+\epsilon$ (due to Arya et al. [6]) and 1.5 (due to Byrka [16]) respectively. Also of interest to us is the best known Lagrangean multiplier preserving approximation algorithm for the facility location problem with performance guarantee of 2, which is due to Jain et al. [41]. The best currently known approximation algorithm for the k-MST problem has a performance guarantee of 2 (Garg [31]).

There has been a lot of previous work on incremental approximation algorithms, but it was usually done on a problem-by-problem basis. Mettu and Plaxton [48] introduce the incremental k-median problem, and give a 29.86-competitive algorithm for it. Their algorithm runs in near linear time and their argument also applies when the distances satisfy a weaker version of triangle inequality, yielding an $O(1)$-competitive solution for the incremental k-means problem. Plaxton [55] introduces the incremental facility location problem and gives a $(4 + \epsilon)\alpha$-competitive algorithm for it, given any α-approximation algorithm for the uncapacitated facility location problem. This yields a $(6 + \epsilon)$-competitive algorithm for incremental facility location using the 1.5-approximation algorithm of [16] for uncapacitated facility location. González [33] gives a 2-approximation algorithm for the k-center problem, which is also a 2-competitive algorithm for the incremental k-center problem. Implicit in the work of Charikar, Chekuri, Feder, and Motwani [17] on incremental clustering are a deterministic 8-competitive algorithm and a randomized $2e$-competitive algorithm for the hierarchical k-center problem. Dasgupta and Long [26] explicitly introduce the idea of finding competitive hierarchical clusterings, and derive the same bounds as above for the hierarchical k-center problem. Plaxton [55] gives an 8α-competitive algorithm for the hierarchical k-median problem, given an α-competitive algorithm for the incremental k-median problem. Using the algorithm of Mettu and Plaxton [48] gives a 238.88-competitive algorithm for the hierarchical k-median problem. The work of [55] also includes an $O(1)$-competitive algorithm for the hierarchical k-means problem. Implicit in work on the minimum latency problem is a number of different algorithms for the incremental k-MST problem; given an α-approximation algorithm for the k-MST problem, the work of Blum et al. [13] yields a 4α-competitive algorithm, while a randomized $e\alpha$-competitive algorithm is implicit in Goemans and Kleinberg [32]. Other work on incremental approximation algorithms includes incremental flow (Hartline and Sharp [36]) and incremental bin packing (Codenotti, De Marco, Leoncini, Mantangero, Santini [25]).

Independently Chrobak, Kenyon, Noga, and Young [21] also discovered the same $(24 + \epsilon)$-competitive deterministic and $(6e + \epsilon)$-competitive randomized algorithms as ours for the incremental k-median problem. They also consider the incremental version of a median problem in which the goal is to minimize the number of medians required to satisfy a given cost constraint. These results are derived by a reduction from a new problem, which they call online bribery, for which tight upper and lower bounds are established. Chrobak et al. [21] also extend their work to fractional k-medians, approximately metric distance functions, which include the k-means objective, and bicriteria approximations.

2.2 A general framework for incremental optimization

In this section, we present a general framework for incremental optimization (Section 2.2.1), a generic approximation algorithm for incremental optimization problems that lie within this framework (Section 2.2.2) and an alternate view of the generic algorithm useful for getting a better competitive ratio (Section 2.2.3).

2.2.1 Problem definitions

The problems we consider in this chapter are all minimization problems and share the following characteristics. Each optimization problem Π can be specified by a quadruple $\langle U, \mathsf{ben}, \mathsf{cost}, p\rangle$, where U is a set of feasible solutions, $\mathsf{ben} : U \to \mathcal{R}$ and $\mathsf{cost} : U \to \mathcal{R}$ are *benefit* and *cost* functions, respectively, and the goal is to seek a solution S that minimizes $\mathsf{cost}(S)$ subject to the condition $\mathsf{ben}(S) \geq p$. We refer to Π as an *offline problem* to distinguish it from its incremental version, which we now define. We introduce a binary relation $\preceq$, which induces a partial order on U, i.e., $\langle U, \preceq\rangle$ is a *poset*. Throughout this chapter, we focus on benefit and cost functions that are nonnegative and monotonically non-decreasing with respect to the partial order; that is, if $S \preceq S'$, then $\mathsf{ben}(S) \leq \mathsf{ben}(S')$ and $\mathsf{cost}(S) \leq \mathsf{cost}(S')$. We also assume that the empty set $\emptyset$ is a feasible solution with cost 0; that is, $\emptyset \in U$ and $\mathsf{cost}(\emptyset) = 0$. (The element $\emptyset$ is also a bottom element of $\preceq$ in all the problem formulations considered in this chapter.)

The incremental version of Π is specified by the quadruple $\langle U, \preceq, \mathsf{ben}, \mathsf{cost}\rangle$ and seeks a *chain* $\mathcal{C}$ of $\langle U, \preceq\rangle$. Define the competitive ratio of $\mathcal{C}$ as

$$\sup_{0<p\leq B_{max}} \frac{\mathsf{cost}(\pi(\mathcal{C}, p))}{\mathsf{cost}(\mathsf{Opt}(p))},$$

where $B_{max} = \max_{S \in U} \mathsf{ben}(S)$ is the maximum benefit achieved by a feasible solution, $\pi(\mathcal{C}, p)$ denotes the smallest indexed element of $\mathcal{C}$ whose benefit is at least p, and $\mathsf{cost}(\mathsf{Opt}(p))$ is the cost of an optimal solution for the offline problem for benefit p, namely $\langle U, \mathsf{ben}, \mathsf{cost}, p \rangle$.

Definition 1 *An α-approximation algorithm for the problem Π finds a solution S in U for every given benefit p such that $\mathsf{cost}(S) \leq \alpha \cdot \mathsf{cost}(\mathsf{Opt}(p))$ and $\mathsf{ben}(S) \geq p$. Usually such problems have the set of integers from 1 to B_{max} as the range of benefit function.*

2.2.2 A generic incremental approximation algorithm

The core of each of our approximation algorithms for incremental optimization problems is a subroutine for augmenting a given solution to achieve a certain benefit. In this section, we present a sufficient condition for the existence of such an augmentation. By repeatedly invoking this augmentation subroutine (which is specific to the particular problem), we show how to derive a sequence that has a good competitive ratio. We begin by defining the augmentation property.

Definition 2 *(γ, δ)-Augmentation: We say that the (γ, δ)-augmentation property holds for reals $\gamma, \delta \geq 0$ if for every solution S of U and every real $p \leq B_{max}$, there exists an augmentation S' such that*

1. *$S \preceq S'$.*
2. *$\mathsf{cost}(S') \leq \gamma \mathsf{cost}(S) + \delta \mathsf{cost}(\mathsf{Opt}(p))$.*
3. *$\mathsf{ben}(S') \geq p$.*

Let $\mathsf{Augment}(S, p, \gamma, \delta)$ *denote a subroutine that computes such an augmentation. For efficiency reasons, we also introduce a companion subroutine* $\mathsf{CostBound}(S, p, \gamma, \delta)$ *which returns a bound on the cost of* $\mathsf{Augment}(S, p, \gamma, \delta)$*. In particular, for every feasible solution* S *and benefit* p*, we have*

$$\mathsf{cost}(\mathsf{Augment}(S, p, \gamma, \delta)) \leq \mathsf{CostBound}(S, p, \gamma, \delta) \leq \gamma \mathsf{cost}(S) + \delta \mathsf{cost}(\mathsf{Opt}(p)).$$

We now present two generic incremental optimization algorithms, given an augmentation subroutine. One is deterministic, while the other is randomized. Since these two algorithms share the same structure, differing only in the parameter setting (the Initialization step below), they are shown together. In the subsequent sections, we show that for each of the problems we consider in this chapter, the augmentation subroutine can be implemented using an approximation algorithm to the offline optimization problem for suitable choices of γ and δ.

Remark 1 *For some applications discussed in this chapter, most notably the incremental and hierarchical median problems, the poset induced by the partial order is, in fact, a ranked poset; that is, every maximal chain in the poset is of the same length. For these problems, we can replace the chain* $\mathcal{C}$ *that is output by the above incremental algorithm by any maximal chain that contains* $\mathcal{C}$*, without increasing the competitive ratio.*

Theorem 1 *Assume that for any* $S \in U$*,* $\mathsf{cost}(S)$ *is either zero or at least one. If* (γ, δ)*-Augmentation holds for reals* $\gamma \geq 1$*,* $\delta \geq 1$*, then (i)* $\textsc{IncApprox}(\gamma, \delta)$ *(Deterministic) computes an incremental solution with competitive ratio* $4\gamma\delta$*; (ii)* $\textsc{IncApprox}(\gamma, \delta)$ *(Randomized) computes an incremental solution with competitive ratio* $\min_\mu \frac{\delta(\mu-1)}{(1-\gamma/\mu)\ln \mu}$*, which equals* $e\delta$*, when* $\gamma = 1$*.*

Algorithm 1: INCAPPROX(γ, δ)

```
1. Initialization:
      $S_0 = \mathsf{Augment}(\emptyset, q, \gamma, \delta)$, where $q$ is the largest value for which
      $\mathsf{CostBound}(\emptyset, q, \gamma, \delta) = 0$. $C_0 = \max\{1, \mathsf{cost}(S_0)\}$.
   1D: (Deterministic) $i = 0$, $\beta = 2\gamma$, $\beta_0 = \beta$.
   1R: (Randomized) $i = 0$, $\beta$ is the minimizer of $\frac{\beta-1}{(1-\gamma/\beta)\ln\beta}$, $\beta_0 = \beta^X$, where $X$ is
       uniform from $[0, 1)$.
2. Iteration $i$: $S_{i+1} = \mathsf{Augment}(S_i, p, \gamma, \delta)$, where $p$ is the largest value for which
   $\mathsf{CostBound}(S_i, p, \gamma, \delta)$ is at most $\beta_0\beta^i C_0$.
3. Termination: If $\mathsf{ben}(S_{i+1}) \neq B_{max}$, $i \leftarrow i + 1$, go to step 2; Otherwise, return sequence
   $S_1, \cdots, S_{i+1}$.
```

Proof: Fix a real $p \leq B_{max}$. Let S^* denote an optimal solution for the offline instance with benefit p. We consider two cases.

If $\mathsf{cost}(S^*) = 0$ then $p \leq \mathsf{ben}(S_0)$ by the maximality of $\mathsf{ben}(S_0)$ and the fact that $\mathsf{CostBound}(\emptyset, p, \gamma, \delta) = 0$. In this case we have found a solution S_0 such that $\mathsf{cost}(S^*) = \mathsf{cost}(S_0) = 0$ and $\mathsf{ben}(S^*) \leq \mathsf{ben}(S_0)$, and the claim of the theorem holds.

The remainder of the proof concerns the case where $\mathsf{cost}(S^*) \neq 0$. We then have $\mathsf{cost}(S^*) \geq 1$. Since either $\mathsf{cost}(S_0) = 0$ or $S_0 = \emptyset$, we also have $\mathsf{cost}(S^*) \geq \mathsf{cost}(S_0)$. Therefore $\mathsf{cost}(S^*) \geq C_0$. Let k be the smallest integer such that $\frac{\delta\mathsf{cost}(S^*)}{1-\gamma/\beta} \leq \beta_0\beta^k C_0$. We note that $k \geq 0$ since

$$\frac{\delta\mathsf{cost}(S^*)}{1-\gamma/\beta} > \mathsf{cost}(S^*) \geq \beta_0 C_0/\beta.$$

We now argue that $\mathsf{cost}(\mathsf{Augment}(S_k, p, \gamma, \delta))$ is at most $\beta_0\beta^k C_0$. We consider two cases: $k = 0$ and $k > 0$. If $k = 0$, then we obtain

$$\begin{aligned}\mathsf{cost}(\mathsf{Augment}(S_0, p, \gamma, \delta)) &\leq \gamma\mathsf{cost}(S_0) + \delta\mathsf{cost}(S^*)\\ &\leq \gamma C_0 + \beta_0(1 - \gamma/\beta)C_0\\ &\leq C_0\beta_0.\end{aligned}$$

For $k > 0$, we use the following property that is enforced by each iteration of IncApprox: for $j \geq 1$, $\mathsf{cost}(S_j) \leq \beta_0\beta^{j-1}C_0$. Applying the preceding inequality with $j = k$, we obtain $\mathsf{cost}(S_k) \leq \beta_0\beta^{k-1}C_0$. We now derive

$$\begin{aligned}\mathsf{cost}(\mathsf{Augment}(S_k, p, \gamma, \delta)) &\leq \gamma\mathsf{cost}(S_k) + \delta\mathsf{cost}(S^*)\\ &\leq \gamma\beta_0\beta^{k-1}C_0 + \beta_0\beta^k C_0 \cdot (1 - \gamma/\beta)\\ &\leq C_0\beta_0\beta^k.\end{aligned}$$

We now establish a bound on the competitive ratios of the two versions of the algorithm using the solution S_{k+1}, which has benefit at least p and cost at most $\beta_0\beta^k C_0$.

Deterministic: By the minimality of k, we lower bound $\mathsf{cost}(S^*)$ by $\frac{\beta_0\beta^k C_0}{\beta} \cdot \frac{1-\gamma/\beta}{\delta}$ and obtain the following upper bound on the competitive ratio of $\mathcal{C}$.

$$\begin{aligned}\mathsf{cost}(\pi(\mathcal{C}, p))/\mathsf{cost}(\mathsf{Opt}(p)) &\leq \mathsf{cost}(S_{k+1})/\mathsf{cost}(\mathsf{Opt}(p))\\ &\leq \frac{\beta_0\beta^k}{\frac{\beta_0\beta^k}{\beta} \cdot \frac{1-\gamma/\beta}{\delta}} = \beta^2\delta/(\beta - \gamma).\end{aligned}$$

The above bound is minimized when $\beta = 2\gamma$, thus yielding a $4\delta\gamma$ competitive ratio.

Randomized: In the randomized algorithm, β_0 is a random variable β^X, where X is uniform in $[0, 1)$. Let Y be the random variable $\log_\beta(\beta_0\beta^k C_0 \frac{1-\gamma/\beta}{\delta\mathsf{cost}(S^*)})$.

We now argue that Y is uniform in $[0, 1)$. Letting C equal $\log_\beta(\frac{\delta \mathsf{cost}(S^*)}{C_0(1-\gamma/\beta)})$, we obtain $Y = k - (C - X)$. On the other hand, it follows from the definition of k that $k = \lceil C - X \rceil$. Since X is chosen uniformly at random in $[0, 1)$, so is Y.

Thus, the expectation of the ratio $\beta_0 \beta^k \frac{(1-\gamma/\beta)C_0}{\delta \mathsf{cost}(S^*)}$ is $\int_0^1 \beta^y dy = \frac{\beta - 1}{\ln \beta}$. We conclude that the competitive ratio is at most

$$\begin{aligned} E\left[\frac{\beta_0 \beta^k C_0}{\mathsf{cost}(S^*)}\right] &= E\left[\frac{\beta_0 \beta^k}{\frac{\delta \mathsf{cost}(S^*)}{C_0(1-\gamma/\beta)}}\right] \cdot \frac{\delta}{1-\gamma/\beta} \\ &= \frac{\delta(\beta - 1)}{(1-\gamma/\beta)\ln \beta}. \end{aligned}$$

We select β to minimize the above bound. In particular, with $\gamma = 1$, we set $\beta = e$, obtaining a ratio of $e\delta$. □

Remark 2 *The assumption about the cost of any feasible solution in the statement of Theorem 1 can be easily enforced (by scaling costs) for all problems studied in this chapter, with the exception of a special case of the facility location problem, which we handle separately in Section 2.3.3.*

We now derive a polynomial the upper bound on the running time of INCAPPROX under the assumption that the range of the benefit function is the set of integers, which is true for all problems studied in this chapter except the incremental facility location problem. We establish the polynomial running time of INCAPPROX for incremental facility location separately in Section 2.3.3.

The running time of INCAPPROX is dominated by the calls to the augmentation subroutine. The number of calls made to the augmentation subroutine in each iteration (including the initialization step) depends on the particular search technique we use to find a maximum benefit augmented solution within a certain cost bound. One simple method is to let the CostBound function simply return the cost of the augmented solution and to perform a linear search through all benefit values. Then the number of calls made during an iteration is at most B_{max}. Since the number of iterations is at most $\log_\beta \frac{\mathsf{Maxcost}}{\beta}$, we obtain an upper bound of $O(B_{max} \log_\beta \frac{\mathsf{Maxcost}}{\beta})$ calls to the augmentation subroutine. For all the problems with integer benefits that we consider in this chapter, B_{max} is at most the size of the problem instance, and hence the running time of INCAPPROX is bounded by a polynomial in the size of the instance.

We can give a more efficient implementation by replacing the above linear search by binary search if, for all S, CostBound(S, p, γ, δ) is a monotonically non-decreasing function of p. (We can define such a CostBound subroutine for all the problems we study in this chapter.) Given this monotonicity property, we can perform a binary search on benefit values to find the maximum benefit augmented solution that has cost at most $\beta_0\beta^i$, according to CostBound. The number of calls made in any iteration is then $O(\log B_{max})$, yielding an upper bound of $O(\log_\beta \frac{\mathsf{Maxcost}}{\beta} \log B_{max})$ total calls to the augmentation subroutine.

2.2.3 An alternate view of the generic incremental approximation algorithm

This alternate view of the algorithm offers better understanding of the algorithm when the range of the benefit function is the set of positive integers. In the above algorithm, we use the augmentation subroutine as a black box. For the incremental problems discussed in this chapter, when we implement this subroutine, we almost always take two elements S_1, S_2 with $\mathsf{ben}(S_1) < \mathsf{ben}(S_2)$, where S_2 is usually a good approximate solution to $\mathsf{Opt}(\mathsf{ben}(S_2))$, and create another element S such that $S_1 \preceq S$ and $\mathsf{ben}(S) \geq \mathsf{ben}(S_2)$. With this implementation detail in mind, a high-level and equivalent view of the above algorithm is the following: Identify a suitable set of benefit values such that the corresponding (good) approximate solutions' costs lie in different *buckets*, where the bucket size increases geometrically. One then constructs a chain out of these solutions iteratively using the augmentation subroutine. This alternate approach is explained in more detail in this section as it helps us give better approximation ratios for some problems.

Definition 3 *(γ, δ')-Nesting: We say that the (γ, δ')-nesting property holds for reals $\gamma, \delta' \geq 0$ if for any two solutions S_1 and S_2 of U with $\mathsf{ben}(S_1) < \mathsf{ben}(S_2)$, there exists a solution S such that*

1. *$S_1 \preceq S$.*
2. *$\mathsf{cost}(S) \leq \gamma \mathsf{cost}(S_1) + \delta' \mathsf{cost}(S_2)$.*
3. *$\mathsf{ben}(S) \geq \mathsf{ben}(S_2)$.*

Let $\mathsf{Nesting}(S_1, S_2, \gamma, \delta')$ denote a subroutine that computes such a solution S.

Remark 3 *Note that the* Augment *and the* Nesting *subroutine are equivalent. The only minor difference is that the* Auugment *subroutine takes a solution and a particular benefit as arguments to find a solution with benefit no less than given benefit whereas the* Nesting *subroutine takes two solutions as arguments and finds a solution with benefit no less than the benefit of the second solution given.*

Algorithm 2: $\textsc{AltIncApprox}(\gamma, \delta', \alpha)$

1. Initialization:

   1D: (Deterministic) $i = 1$, $S_0 = \emptyset$, $\beta = 2\gamma$, $\beta_0 = 1$.

   1R: (Randomized) $i = 1$, $S_0 = \emptyset$, β is the minimizer of $\frac{\beta-1}{(1-\gamma/\beta)\ln\beta}$, $\beta_0 = \beta^X$, where X is uniform from $[0, 1)$.

2. Use an α-approximation algorithm to compute approximate solutions $V_1, V_2, \ldots V_{B_{max}}$ for benefit values $1, 2, \ldots, B_{max}$ respectively.
3. Bucketing: Order these solutions according to their cost into buckets of form $[0, 0]$, $(\beta_0, \beta_0\beta]$, $(\beta_0\beta, \beta_0\beta^2]$, $\ldots$, $(\beta_0\beta^{k-1}, \beta_0\beta^k]$, $\ldots$
4. Pick the solution with highest benefit from each of these non-empty buckets. Let these solutions be $\overline{V}_1, \overline{V}_2, \ldots, \overline{V}_r = V_{B_{max}}$ respectively.
5. Iteration i: $S_i = \mathsf{Nesting}(S_{i-1}, \overline{V}_i, \gamma, \delta')$,
6. Termination: If $\mathsf{ben}(S_i) \neq B_{max}$, $i \leftarrow i + 1$, go to step 5; Otherwise, return sequence $S_1, \cdots, S_i$.

Theorem 2 *If (γ, δ')-nesting holds for reals $\gamma \geq 1$, $\delta' > 0$, and an α-approximation algorithm exists for the problem, then (i) $\textsc{AltIncApprox}(\gamma, \delta', \alpha)$ (Deterministic) computes an incremental solution with competitive ratio $4\gamma\delta'\alpha$; $\textsc{AltIncApprox}(\gamma, \delta', \alpha)$ (Randomized) computes an incremental solution with competitive ratio $\min_\beta \frac{\delta'\alpha(\beta-1)}{(1-\gamma/\beta)\ln\beta}$, which equals $e\delta'\alpha$, when $\gamma = 1$.*

Proof: Fix a $p \leq B_{max}$. If $p \leq \mathsf{ben}(S_0)$, then the solution $S_0 = \emptyset$ provides benefit at least p with least cost, which implies $\mathsf{cost}(\pi(\mathcal{C}, p))$ equals $\mathsf{cost}(\mathsf{Opt}(p))$, establishing the desired claim for this case.

In the remainder, we assume that $p > \mathsf{ben}(S_0)$. Let $i \geq 1$ be such that $\mathsf{ben}(S_{i-1}) < p \leq \mathsf{ben}(S_i)$. Let the solution $\overline{V}_i$ be from the bucket $(M/\beta, M]$. So $\mathsf{cost}(\overline{V}_i) \leq M$, $\mathsf{cost}(\overline{V}_{i-1}) \leq M/\beta$ and so on.

Deterministic case:

$$\begin{aligned}
\mathsf{cost}(\pi(\mathcal{C}, p)) &= \mathsf{cost}(S_i) \\
&\leq \delta' \mathsf{cost}(\overline{V}_i) + \gamma \mathsf{cost}(S_{i-1}) \\
&\leq \delta' \mathsf{cost}(\overline{V}_i) + \gamma\delta' \mathsf{cost}(\overline{V}_{i-1}) + \gamma^2\delta' \mathsf{cost}(S_{i-2}) \\
&\leq \delta' \mathsf{cost}(\overline{V}_i) + \gamma\delta' \mathsf{cost}(\overline{V}_{i-1}) + \gamma^2\delta' \mathsf{cost}(\overline{V}_{i-2}) + \ldots
\end{aligned}$$

$$\begin{aligned}
&\leq \delta' M \left(1 + \frac{\gamma}{\beta} + \frac{\gamma^2}{\beta^2} + \ldots\right) \\
&\leq \frac{\delta' M}{1 - \frac{\gamma}{\beta}} \qquad (2.1) \\
&\leq \frac{\delta' \beta}{1 - \frac{\gamma}{\beta}} \cdot \mathsf{cost}(V_p) \\
&\leq \frac{\delta' \beta \alpha}{1 - \frac{\gamma}{\beta}} \cdot \mathsf{cost}(\mathsf{Opt}(p))
\end{aligned}$$

The second inequality follows from (γ, δ')-Nesting inequality $\mathsf{cost}(S_{i-1}) \leq \delta' \mathsf{cost}(\overline{V}_{i-1}) + \gamma \mathsf{cost}(S_{i-2})$ and the penultimate equation follows from the fact that $\mathsf{cost}(V_p)$ lies between M/β and M and so $M \leq \beta \mathsf{cost}(V_p)$. This ratio is minimized when $\beta = 2\gamma$ which gives a $4\gamma\delta'\alpha$-competitive algorithm.

Randomized case: Since β_0 is a random variable β^X where X is uniform in $[0, 1)$, it follows that $M/\mathsf{cost}(V_p)$ is a random variable β^Y, where Y is uniform

$[0,1)$. From Equation 2.1,

$$\begin{aligned}
E\left[\mathsf{cost}(\pi(\mathcal{C},p))\right] &\leq E\left[\frac{\delta' M}{1-\frac{\gamma}{\beta}}\right] \\
&\leq \left(\frac{\delta'\mathsf{cost}(V_p)}{1-\frac{\gamma}{\beta}}\right)\cdot E\left[\frac{M}{\mathsf{cost}(V_p)}\right] \\
&\leq \left(\frac{\delta'\mathsf{cost}(V_p)}{1-\frac{\gamma}{\beta}}\right) E[\beta^Y] \\
&\leq \left(\frac{\delta'\mathsf{cost}(V_p)}{1-\frac{\gamma}{\beta}}\right)\cdot \frac{\beta-1}{\ln\beta} \\
&\leq \left(\frac{\delta'(\beta-1)\alpha}{\left(1-\frac{\gamma}{\beta}\right)\ln\beta}\right)\cdot \mathsf{cost}(\mathsf{Opt}(p))
\end{aligned}$$

When $\gamma = 1$, the ratio is minimized at $\beta = e$ which gives an $e\delta'\alpha$-competitive algorithm. □

The running time of ALTINCAPPROX is dominated by the calls to the approximation algorithm which is done B_{max} times. Then we call the nesting subroutine r times which is bounded by B_{max}. So the algorithm requires B_{max} calls to the approximation algorithm and at most B_{max} calls to the nesting subroutine.

2.3 Applications

In this section, we apply our framework of Section 2.2 to incremental versions of several classical optimization problems.

2.3.1 The incremental k-MST problem

Given a complete graph $G = (V, E)$, $|V| = n$, with metric cost function $w : E \to Q^+$ and a root $r \in V$, the (rooted) *k-MST problem* seeks a minimum-cost subgraph of G that spans at least k vertices, including r. In the incremental k-MST problem, we seek a sequence of $n-1$ edges of E, $e_1, e_2, \ldots, e_{n-1}$ such that for any $k \in [2, n]$, the first $k-1$ edges of the sequence span k vertices including r. For each k, consider the ratio of the sum of the cost of the first $k-1$ edges to the cost of an optimal k-MST of G that covers r. The goal of incremental k-MST is to seek a sequence of edges that minimizes the maximum of this ratio, over all k.

In our framework, U is the set of all connected subgraphs of G that contain r and $\preceq$ is the $\subseteq$ relation of the edge subsets. The benefit of a solution is the number of vertices it spans, and the cost is the sum of the edge weights.

Lemma 1 *There exists a $(1, 1)$-augmentation for the k-MST problem, and a $(1, \alpha)$-augmentation that can be implemented in polynomial time, where $\alpha = 2$. Also a $(1, 1)$-nesting exists and can be efficiently implemented for the k-MST problem.*

Proof: The augmentation operation $\mathsf{Augment}$ is straightforward. Given any component $S \subseteq E$ spanning r, and $k \le n$, if it already contains at least k vertices, we are done. Otherwise, let S^* be an optimum solution to the rooted k-MST problem. We can take the edges of S and a subset of edges of S^* to connect the vertices in S^* to those of S. This is possible since both the S and S^* contains the root vertex. This yields a component S' spanning at least $k = |S^*|$ vertices with cost bounded by $\mathsf{cost}(S) + \mathsf{cost}(S^*)$. The companion $\mathsf{CostBound}$ function returns $\mathsf{cost}(S) + \mathsf{cost}(S^*)$.

Similarly, in order to implement the augmentation efficiently, we use the polynomial-time 2-approximation algorithm [31] to obtain a solution S_1 to the k-MST problem, and connect the newly discovered vertices to S. This yields a component S' spanning at least k vertices with cost at most $\mathsf{cost}(S) + \mathsf{cost}(S_1)$ which is at most $\mathsf{cost}(S) + 2\mathsf{cost}(S^*)$. To obtain a monotonic cost bound function, we have CostBound return $\mathsf{cost}(S)+\mathsf{cost}(S_1)$. Since the cost of the k-MST solution returned by the algorithm of [31] is monotonically increasing with k, CostBound is mononotic with respect to the benefit parameter.

The existence of $(1, 1)$-Nesting and efficient implementation follows by taking S and S^* to be S_1 and S_2 which are the inputs to the nesting subroutine.
□

Theorem 3 *There exists a solution to incremental k-MST problem with competitive ratio 4. A deterministic solution with competitive ratio 4α and a randomized solution with competitive ratio $e\alpha$ can be computed efficiently, where $\alpha = 2$.*

Proof: Immediate from Lemma 1 and Theorem 1. □

We note that this computation of an 8-competitive incremental MST is implicit in the work of Blum et al. [13].

2.3.2 Incremental and hierarchical median problems

The incremental k-median problem

Given the locations of a set F of $|F| = n_f$ facilities and a set C of $|C| = n_c$ clients in a metric space, *the k-median problem* is that of finding a set of k facilities to *open* such that the sum of the distances of the clients to the nearest open facility is minimized. The discussions in this section can be extended to the case when clients have demands and the problem is to find a set of k facilities so as to minimize the demand-weighted distances of clients to the nearest open facility. Let c_{ij} denote the distance between any two locations i and j. In the incremental k-median problem, we seek an ordering of the facilities. For each k, consider the ratio of the cost of opening the first k facilities in the ordering to the cost of an optimal k-median solution. The goal is to find an ordering that minimizes the maximum of this ratio over $k = 1, \ldots, n_f$. An algorithm for the problem is said to be *α-competitive* if the maximum of the ratio over all k is no more than α. This value α is called the *competitive ratio* of the algorithm.

We model the incremental median problem using our framework of Section 2.2 by the quadruple $\langle U, \preceq, \mathsf{ben}, \mathsf{cost} \rangle$. The set $U = 2^F$ is the set of all feasible solutions, each solution represented by the set of open facilities. The binary relation is given as $S_1 \preceq S_2$ iff $S_1 \supseteq S_2$, $\mathsf{ben}(S)$ equals $n_f - |S|$, and $\mathsf{cost}(S)$ is the cost of connecting the clients to their nearest facilities in S. The output of our incremental approximation algorithm is a chain of subsets of the facilities, where each chain element (subset of facilities) is a subset of the previous element. As shown in Theorem 4 below, the desired sequence of facilities for the incremental median problem is simply a concatenation of the differences between consecutive sets of this chain, presented in reverse order. The main claim of the following lemma is implicit in [42] and [22].

Lemma 2 *There exists a $(1, 2)$-augmentation for the incremental median problem. A $(1, 2\alpha)$-augmentation can be efficiently implemented, where $\alpha = 3 + \epsilon$. Also a $(1, 2)$-nesting exists and can be efficiently implemented.*

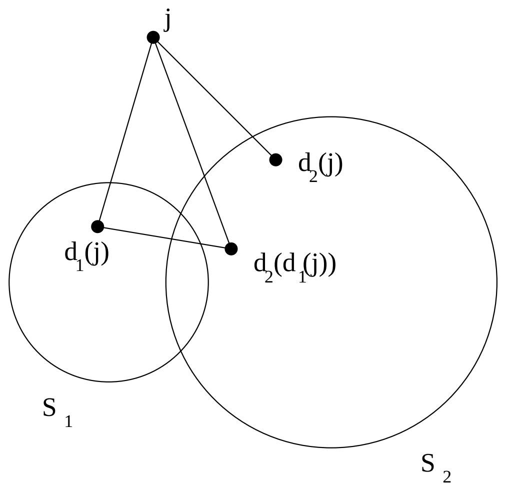

Figure 2.1: Reassigning clients in the proof of Lemma 2

Proof: Let S_2 ($\mathsf{ben}(S_2) < p$) be a set of facilities. We would like to augment it to get a benefit of at least p. Let S_1 be a set of facilities with benefit p. According to the definition, $|S_2| > |S_1|$. We aim to find a subset S such that $S_2 \preceq S$, i.e., $S \subset S_2$, and $|S| \leq |S_1|$.

For any location (client or facility) j, let $d_1(j)$ (resp., $d_2(j)$) be the closest facility to j in S_1 (resp., S_2). For any client j let us bound the distance $c_{j,d_2(d_1(j))}$ (see Figure 2.1):

$$\begin{aligned} c_{j,d_2(d_1(j))} &\leq c_{j,d_1(j)} + c_{d_1(j),d_2(d_1(j))} \\ &\leq c_{j,d_1(j)} + c_{d_1(j),d_2(j)} \\ &\leq c_{j,d_1(j)} + c_{j,d_1(j)} + c_{j,d_2(j)} \\ &= 2c_{j,d_1(j)} + c_{j,d_2(j)}, \end{aligned}$$

where the second inequality follows since $d_2(d_1(j))$ is the closest median in S_2 to $d_1(j)$. Define $S = \{d_2(i) : i \in S_1\}$; that is, S is the set of facilities in S_2 that are closest to the facilities in S_1. Let $d(j)$ be the closest facility in S for a location j.

For any client j, $c_{j,d(j)} \leq c_{j,d(d_1(j))} = c_{j,d_2(d_1(j))} \leq 2c_{j,d_1(j)} + c_{j,d_2(j)}$. Summing this over all clients, we obtain $\text{cost}(S) \leq \text{cost}(S_2) + 2\text{cost}(S_1)$. Note that $S \subset S_2$ and $|S| \leq |S_1|$. This proves the existence and efficient implementation of a $(1, 2)$-nesting.

Using an optimum solution (resp., α-approximate solution [6]) to the k-median problem for S_1 proves the existence (resp., efficient implementation) of the $(1, 2)$-augmentation (resp., $(1, 2\alpha)$-augmentation). □

Theorem 4 *There exists a solution to the incremental median problem with competitive ratio* 8*. A deterministic solution with competitive ratio* 8α *and a randomized solution with competitive ratio* $2e\alpha$ *can be computed efficiently, where* $\alpha = 3 + \epsilon$*.*

Proof: The existence and computability of chains of $\langle U, \preceq \rangle$ with the desired competitive ratios follow immediately from Lemma 2 and Theorem 1. To convert a given chain $\mathcal{C}$ of facility sets into a sequence of medians, we simply generate a maximal chain containing $\mathcal{C}$ and concatenate the differences between consecutive sets of this chain in reverse order. By the definition of competitive ratio (see Section 2.2.1), the competitive ratio of the chain is at least that of the median sequence, thus completing the proof of the theorem. □

The hierarchical k-median problem

We define *an assignment* of a k-median solution as function from clients to facilities that assigns each client to an open facility in the solution. In the hierarchical k-median problem, we give an ordering of facilities along with assignments $a_1, \ldots, a_{n_f}$ such that the assignment a_k assigns clients only to the first k facilities in the ordering; this corresponds to a clustering with k clusters. To ensure that the clusterings are formed by merging pairs of clusters, we require that for any two assignments a_k and a_{k-1} that a_{k-1} can be obtained from a_k by reassigning all the clients assigned to the kth facility in the ordering to a single facility earlier in the ordering. Now consider the ratio of the cost of assignment a_k to the cost of an optimal k-median solution. The goal of the problem is to find an ordering of facilities and a valid sequence of assignments so as to minimize the maximum of this ratio over all $k = 1, \ldots, n_f$. An algorithm for the problem is said to be *α-competitive* if the maximum of the ratio over all k is no more than α.

We show how to cast the hierarchical median problem into our incremental optimization framework. A solution to the k-median problem is represented as a pair (S, a) containing a subset S of facilities and an assignment a of clients to facilities in S. In the incremental k-median problem, the assignment function assigns each client to its nearest available facility. This cannot be assumed for the hierarchical median problem. For any i in S, let $a^{-1}(i)$ be the set of clients assigned to i by a. Given a solution (S, a), we say that a is *locally-optimal* for S if for all i in S, assigning all clients in $a^{-1}(i)$ to any other single facility in S will not decrease the total cost. We adopt the convention that if the assignment is omitted, the default assignment is to assign each client to its nearest available facility.

In the quadruple $\langle U, \preceq, \mathsf{ben}, \mathsf{cost}\rangle$, we let U be the set of all pairs (S, a) such that a is locally-optimal in (S, a). It is easy to see that U includes all optimal k-median solutions, for all values of k. Let $\mathsf{cost}(S, a)$ represents the cost of assigning the clients to the facilities in S according to the assignment a whereas $\mathsf{cost}(S)$ represents the cost of assigning clients to the nearest facility in S.

Definition 4 *We say two solution pairs* (S_1, a_1) *and* (S_2, a_2) *in* U *are* nested *if*

1. $S_1 \subset S_2$;
2. $\forall j \in C$, *if* $a_2(j) \in S_1$ *then* $a_1(j) = a_2(j)$;
3. $\forall j, k \in C$, *if* $a_2(j) = a_2(k)$ *then* $a_1(j) = a_1(k)$.

We denote nested solutions by $(S_1, a_1) \subset (S_2, a_2)$.

We define $\preceq$ as $(S_2, a_2) \preceq (S_1, a_1)$ iff $(S_1, a_1) \subset (S_2, a_2)$, the benefit of a solution (S, a) as $n_f - |S|$, and the cost of (S, a) to be the service cost for the clients according to the assignment a. By definition, the benefit function is monotonically non-decreasing with the partial order. The same holds for the cost function since the assignment in any solution is locally optimal. We now develop an incremental approximation for $\langle U, \preceq, \mathsf{ben}, \mathsf{cost}\rangle$ and show that a chain output by this algorithm can be transformed to a hierarchical ordering of solution pairs, with the desired competitive ratio.

We first prove the following lemma which will be useful in deriving the augmentation lemma.

Lemma 3 *Given a set* V_1 *of facilities and a solution* $(V_2, a_2) \in U$ *such that* $V_1 \subseteq V_2$, *we can obtain a solution* $(V_1, a_1) \in U$ *such that* (V_1, a_1) *and* (V_2, a_2) *are nested and* $\mathsf{cost}(V_1, a_1) \le 2\mathsf{cost}(V_2, a_2) + \mathsf{cost}(V_1)$.

Proof: Let $d_1(j)$ denote the nearest median in V_1 to the client j. We define two functions P and Q. The function P maps each facility in $V_2 \setminus V_1$ to the nearest facility in V_1. This is the "parent" function of the hierarchical algorithms of Dasgupta and Long [26] and Plaxton [55]. The function Q maps each facility i in

$V_2 \setminus V_1$ to a facility in V_1, which serves the clients in $a_2^{-1}(i)$ at the least total cost, among all facilities in V_1.

We now create assignment a_1: for any client j, $a_1(j) = a_2(j)$ if $a_2(j) \in V_1$ and $a_1(j) = Q(a_2(j))$ if $a_2(j) \in V_2 \setminus V_1$. It is easy to verify that (V_1, a_1) is locally-optimal and the pairs (V_1, a_1) and (V_2, a_2) are nested. Consider the assignment a, which is defined the same way as a_1 by replacing P for Q. By the definition of Q, it follows that $\mathsf{cost}(V_1, a_1) \leq \mathsf{cost}(V_1, a)$. If we show that $c_{j,a(j)} \leq 2c_{j,a_2(j)} + c_{j,d_1(j)}$ for all clients j, then summing this over all clients in C will give the required result.

If $a_2(j) \in V_1$ then $c_{j,a(j)} = c_{j,a_2(j)} \leq 2c_{j,a_2(j)} + c_{j,d_1(j)}$. If $a_2(j) \in V_2 \setminus V_1$ then

$$\begin{aligned} c_{j,a(j)} = c_{j,P(a_2(j))} &\leq c_{j,a_2(j)} + c_{a_2(j),P(a_2(j))} \\ &\leq c_{j,a_2(j)} + c_{a_2(j),d_1(j)} \\ &\leq 2c_{j,a_2(j)} + c_{j,d_1(j)}. \end{aligned}$$

The second inequality follows from the definition of $P(\cdot)$; the third is due to triangle inequality. □

Lemma 4 *There exists a $(3,2)$-augmentation for the hierarchical median problem. A $(3,2\alpha)$-augmentation can be efficiently implemented, where $\alpha = 3 + \epsilon$.*

Proof: Let the subroutine $\mathsf{Augment}(S, p, \gamma, \delta)$ be called with $S = (V_2, a_2) \in U$ and $\mathsf{ben}(S) < p$. Let V be a solution to the k-median problem with benefit p and let a be the closest facility assignment of clients to V. Using Lemma 2 we can find another set $V_1 \subset V_2$ with $|V_1| = |V|$ such that $\mathsf{cost}(V_1) \leq \mathsf{cost}(V_2) + 2\mathsf{cost}(V)$. Using V_1 and (V_2, a_2) in Lemma 3 we get $\mathsf{cost}(V_1, a_1) \leq \mathsf{cost}(V_1) + 2\mathsf{cost}(V_2, a_2)$.

Since $\mathsf{cost}(V_2) \leq \mathsf{cost}(V_2, a_2)$,

$$\begin{aligned}
\mathsf{cost}(V_1, a_1) &\leq \mathsf{cost}(V_1) + 2\mathsf{cost}(V_2, a_2) \\
&\leq 2\mathsf{cost}(V) + \mathsf{cost}(V_2) + 2\mathsf{cost}(V_2, a_2) \\
&\leq 3\mathsf{cost}(V_2, a_2) + 2\mathsf{cost}(V).
\end{aligned}$$

Using an optimum solution (resp., an α-approximate solution [6]) to the k-median problem for V proves the first (resp., second) assertion. □

Theorem 5 *There exists a solution to the hierarchical median problem with competitive ratio* 24*. A deterministic solution with competitive ratio* 24α *and a randomized solution with competitive ratio* 10.76α *can be computed efficiently, where* $\alpha = 3 + \epsilon$*.*

Proof: The existence and computability of chains of $\langle U, \preceq \rangle$ with the desired competitive ratios follow from Lemma 4 and Theorem 1. To convert a given chain $\mathcal{C}$ into a hierarchical sequence of solution pairs, we return the reverse of a maximal chain that contains $\mathcal{C}$. One can obtain a maximal chain of a given chain as follows: between any consecutive pair of solution pairs (S_0, a_0) and (S, a) such that $|S_0| = |S| + k$, for some $k > 1$, insert solution pairs (S_1, a_1), (S_2, a_2), …, (S_{k-1}, a_{k-1}), where S_i equals $S_{i-1} \setminus \{f\}$ for an arbitrary f in $S_{i-1} \setminus S$ and a_i is identical to a_{i-1} except that all clients assigned to f in a_{i-1} are assigned to a facility in S_i that offers the least service cost. By the definition of competitive ratio (see Section 2.2.1), the competitive ratio of the chain is at least that of the hierarchical sequence, thus completing the proof of the theorem. □

Here we use the ALTINCAPPROX(γ, δ, α) algorithm to give better approximation ratios for the hierarchical k-median problem. We can then take advantage of the fact that two solutions that we wish to nest have costs that are geometrically related. We use α-approximate k-median solutions $V_1, V_2, \ldots, V_{B_{max}}$ and then bucket them according to Step 3 of the algorithm to arrive at solutions $\overline{V}_1, \overline{V}_2, \ldots, \overline{V}_r = V_{B_{max}}$ one from each non-empty bucket. Now we get nested solutions along with their assignments from these solutions using Lemma 2 and Lemma 3 as follows. Let $S_1 = \overline{V}_1$ and a_1 be the assignment of clients to the nearest facility in S_1. S_{i+1} is iteratively obtained using Lemma 2 on S_i and $\overline{V}_{i+1}$ with

$$\mathsf{cost}(S_{i+1}) \leq 2\mathsf{cost}(\overline{V}_{i+1}) + \mathsf{cost}(S_i).$$

Then assignment a_{i+1} on S_{i+1} can be obtained from S_i, a_i and S_{i+1} using Lemma 3 with

$$\mathsf{cost}(S_{i+1}, a_{i+1}) \leq 2\mathsf{cost}(S_i, a_i) + \mathsf{cost}(S_{i+1}).$$

Theorem 6 *There exists a solution to the hierarchical median problem with competitive ratio* 20.71*. A deterministic solution with competitive ratio* 20.71α *and a randomized solution with competitive ratio* 10.03α *can be computed efficiently, where* $\alpha = 3 + \epsilon$*.*

Proof: The nested solutions obtained from the ALTINCAPPROX(γ, δ, α) give the corresponding approximation ratios for the hierarchical k-median problem.

Fix a $p \leq B_{max}$. Let i be such that $\mathsf{ben}(S_{i-1}, a_{i-1}) < p \leq \mathsf{ben}(S_i, a_i)$. Let the solution $\overline{V}_i$ be from the bucket $(M/\beta, M]$. So $\mathsf{cost}(\overline{V}_i) \leq M$, $\mathsf{cost}(\overline{V}_{i-1}) \leq M/\beta$ and so on.

$$\begin{aligned}
\mathsf{cost}(S_i, a_i) &\leq \mathsf{cost}(S_i) + 2\mathsf{cost}(S_{i-1}, a_{i-1}) \\
&\leq \mathsf{cost}(S_i) + 2\mathsf{cost}(S_{i-1}) + 2^2\mathsf{cost}(S_{i-2}, a_{i-2}) \\
&\leq \sum_{j=0}^{i} 2^j \mathsf{cost}(S_{i-j}) \\
&\leq 2\mathsf{cost}(\overline{V}_i) + \mathsf{cost}(S_{i-1}) + \sum_{j=1}^{i} 2^j \mathsf{cost}(S_{i-j}) \\
&= 2\mathsf{cost}(\overline{V}_i) + (1+2)\mathsf{cost}(S_{i-1}) + \sum_{j=2}^{i} 2^j \mathsf{cost}(S_{i-j}) \\
&\leq 2\mathsf{cost}(\overline{V}_i) + (1+2)(2\mathsf{cost}(\overline{V}_{i-1}) + \mathsf{cost}(S_{i-2})) + \sum_{j=2}^{i} 2^j \mathsf{cost}(S_{i-j}) \\
&\leq \sum_{j=0}^{i} \left(\sum_{m=0}^{j} 2^m \right) 2\mathsf{cost}(\overline{V}_{i-j}) \\
&\leq 2 \sum_{j=0}^{i} (2^{j+1} - 1) \frac{M}{\beta^j} \\
&\leq 2M \left(\sum_{j=0}^{\infty} \frac{2^{j+1}}{\beta^j} - \sum_{j=0}^{\infty} \frac{1}{\beta^j} \right) \\
&= 2M \left(\frac{2\beta}{\beta - 2} - \frac{\beta}{\beta - 1} \right) \\
&= \frac{2\beta^2 M}{(\beta-1)(\beta-2)}.
\end{aligned}$$

$$\mathsf{cost}(S_i, a_i) \leq \frac{2\beta^2 M}{(\beta-1)(\beta-2)} \leq \frac{2\beta^3 \mathsf{cost}(V_p)}{(\beta-1)(\beta-2)} \leq \frac{2\beta^3 \alpha}{(\beta-1)(\beta-2)} \mathsf{cost}(\mathsf{Opt}(p)).$$

Optimizing for β gives $\beta = 3 + \sqrt{3}$ which gives a 20.709α-competitive approximation for hierarchical k-median problem. For the randomized algorithm, we obtain

$$E[\mathsf{cost}(S_i, a_i)] \leq \frac{2\beta^2 E[M]}{(\beta-1)(\beta-2)} = \frac{2\beta^2 \mathsf{cost}(V_p)}{(\beta-2)\ln\beta} \leq \frac{2\beta^2\alpha}{(\beta-2)\ln\beta}\mathsf{cost}(\mathsf{Opt}(p))$$

Optimizing this equation for β gives $\beta = 6.355$ which gives a 10.03α-competitive randomized approximation algorithm for the hierarchical k-median problem. □

The incremental k-means problem

This problem is identical to the k-median counterpart, except that the k-means cost function is the the sum of squares of the distances of the clients to their nearest open facility. The set of solutions, their benefit, the binary operator, and the structure of posets are exactly the same as that of the k-median problem.

Lemma 5 *There exists a $(2, 8)$-augmentation for the incremental k-means problem. Given an α-approximation algorithm to the k-means problem, a $(2, 8\alpha)$-augmentation can be computed. Also, a $(2, 8)$-nesting exists and can be efficiently implemented.*

Proof: From the proof of Lemma 2 we know that for every location j, $c_{j,d(j)} \leq 2c_{j,d_1(j)} + c_{j,d_2(j)}$. Here $d(j), d_1(j)$ and $d_2(j)$ are as defined in the proof. Squaring this equation we get

$$\begin{aligned} c^2_{j,d(j)} &\leq 4c^2_{j,d_1(j)} + c^2_{j,d_2(j)} + 4c_{j,d_1(j)}c_{j,d_2(j)} \\ &\leq 4c^2_{j,d_1(j)} + c^2_{j,d_2(j)} + \tau^2 c^2_{j,d_1(j)} + \frac{4}{\tau^2}c^2_{j,d_2(j)} \\ &\leq (4+\tau^2)c^2_{j,d_1(j)} + \left(1 + \frac{4}{\tau^2}\right)c^2_{j,d_2(j)} \end{aligned}$$

for all $\tau > 0$. The result follows by summing the above equation over all clients and taking $\tau = 2$. □

Theorem 7 *There exists a solution to the incremental k-means problem with competitive ratio 64. Given a polynomial-time α-approximation for the k-means problem, a deterministic solution and a randomized solution with competitive ratios 64α and 31.82α, respectively, can also be computed efficiently.*

Proof: This is immediate from Lemma 5 and Theorem 1. For the randomized case we minimize the competitive ratio for the values of τ and β to get a 32.82α competitive algorithm with $\tau = 1.61$ and $\beta = 6.47$. $\square$

The hierarchical k-means problem

This problem again is very similar to the hierarchical k-median problem with a slight change in the cost function. In the hierarchical k-means problem, we give an ordering of facilities along with assignments $a_1, \ldots, a_{n_f}$ such that the assignment a_k assigns clients only to the first k facilities in the ordering. Here as in the hierarchical median problem the assignments are nested. The cost of an assignment a is the sum of squares of the distances of the clients to the facility assigned to that client by the assignment a. Now consider the ratio of the cost of assignment a_k to the cost of an optimal k-means solution. The goal of the problem is to find an ordering of facilities and a valid sequence of assignments so as to minimize the maximum of this ratio over all $k = 1, \ldots, n_f$.

The set of solutions, their benefit, the binary operator, and the structure of posets are exactly the same as that of the hierarchical k-median problem.

Lemma 6 *Given two k-means solutions V_1 and V_2, $V_1 \subset V_2$ and an assignment a_2 on V_2 we can obtain an assignment a_1 on V_1 such that (V_1, a_1) and (V_2, a_2) are nested and* $\mathsf{cost}(V_1, a_1) \leq (4 + \rho^2)\mathsf{cost}(V_2, a_2) + (1 + \frac{4}{\rho^2})\mathsf{cost}(V_1)$ *for all $\rho > 0$.*

Proof: We define the functions d_1 and P and the assignment a_1 exactly as we defined in the proof of Lemma 3. From the proof we know that $c_{j,a_1(j)} \leq 2c_{j,a_2(j)} + c_{j,d_1(j)}$ for all clients j. By squaring this equation, we get

$$\begin{aligned}
c^2_{j,a_1(j)} &\leq 4c^2_{j,a_2(j)} + c^2_{j,d_1(j)} + 4c_{j,a_2(j)}c_{j,d_1(j)} \\
&\leq 4c^2_{j,a_2(j)} + c^2_{j,d_1(j)} + \rho^2 c^2_{j,a_2(j)} + \frac{4}{\rho^2}c^2_{j,d_1(j)} \\
&\leq (4+\rho^2)c^2_{j,a_2(j)} + \left(1+\frac{4}{\rho^2}\right)c^2_{j,d_1(j)}
\end{aligned}$$

for all $\rho > 0$. We arrive at the required result by summing the above equation over all clients. □

Lemma 7 *There exists an $(18, 8)$-augmentation for the hierarchical k-means problem. Given an α-approximation algorithm for the k-means problem, an $(18, 8\alpha)$-augmentation can be computed. Also, an $(18, 8)$-nesting exists and can be efficiently implemented.*

Proof: Given (V_2, a_2), with $\mathsf{ben}(V_2) < p$, let V be a solution to the k-means problem with benefit p. Along the same lines of proof of the Lemma 5 we can find another set $V_1 \subset V_2$ with $|V_1| = |V|$ such that $\mathsf{cost}(V_1) \leq (1 + \frac{4}{\tau^2})\mathsf{cost}(V_2) + (4+\tau^2)\mathsf{cost}(V)$. Using this V_1 and (V_2, a_2) in Lemma 6 we get $\mathsf{cost}(V_1, a_1) \leq (4+\rho^2)\mathsf{cost}(V_2, a_2) + (1 + \frac{4}{\rho^2})\mathsf{cost}(V_1)$. Since $\mathsf{cost}(V_2) \leq \mathsf{cost}(V_2, a_2)$ we get

$$\begin{aligned}
\mathsf{cost}(V_1, a_1) &\leq (4+\rho^2)\mathsf{cost}(V_2, a_2) + \left(1+\frac{4}{\rho^2}\right)\mathsf{cost}(V_1) \\
&\leq (4+\rho^2)\mathsf{cost}(V_2, a_2) + \left(1+\frac{4}{\rho^2}\right)\left(1+\frac{4}{\tau^2}\right)\mathsf{cost}(V_2) \\
&\quad + \left(1+\frac{4}{\rho^2}\right)(4+\tau^2)\mathsf{cost}(V) \\
&= \left[4+\rho^2+\left(1+\frac{4}{\rho^2}\right)\left(1+\frac{4}{\tau^2}\right)\right]\mathsf{cost}(V_2, a_2) \\
&\quad + \left(1+\frac{4}{\rho^2}\right)(4+\tau^2)\mathsf{cost}(V)
\end{aligned}$$

for all $\rho, \tau > 0$. Optimizing for the values of ρ and τ so as to minimize the deterministic competitive ratio for the generalized algorithm gives $\rho = 2\sqrt{2}$ and $\tau = \frac{2}{\sqrt{3}}$. This reduces the inequality to $\mathsf{cost}(V_1, a_1) \leq 18\mathsf{cost}(V_2, a_2) + 8\mathsf{cost}(V)$. Using an optimum solution to k-means problem for V proves the first assertion. Using an α-approximate solution to the k-means problem for V proves the second assertion. □

Theorem 8 *There exists a solution to the hierarchical k-means problem with competitive ratio 576. Given an α-approximation algorithm for the k-means problem, a deterministic and a randomized solution with competitive ratios 576α and 151.1α can be computed.*

Proof: In the deterministic case the competitive ratio is obtained by using the augmentation algorithm in Lemma 7. For randomized case we optimize the competitive ratio of the randomized algorithm for the values of ρ, τ and β to arrive at a competitive ratio of 151.01α with $\rho \simeq 3.15, \tau \simeq 1.01$ and $\beta \simeq 48.1$. The existence and computability of chains of $\langle U, \preceq \rangle$ with the desired competitive ratios follow immediately from Lemma 7 and Theorem 1. We use the same approach followed in the proof of Theorem 5 to convert a given chain of $\mathcal{C}$ of solution pairs to a hierarchical sequence of solution pairs, thus completing the proof of the theorem. □

2.3.3 The incremental facility location problem

This problem was first defined by Plaxton [55], who also gives a $(4+\epsilon)\alpha$ competitive algorithm, where α is the best available approximation factor for the facility location problem. We show that our framework also handles this problem with competitive ratio 4α.

The setting is similar to that of the k-median problem. Here, though there is no restriction on the number of facilities to open, instead there is a facility cost $v(i)$ for opening a facility i in addition to the connection costs between each client and the nearest open facility. Without loss of generality, we assume that the minimum non-zero cost of opening a facility is 1; i.e., $\min\{v(i) : i \in F, v(i) \neq 0\}$ is 1 (otherwise, the facility costs can be scaled appropriately). To define the incremental facility location problem, we introduce a positive scaling factor λ, so that the total cost associated with opening a subset $Y \subseteq F$ is

$$\mathsf{cost}_\lambda(C, Y) = \lambda \cdot \sum_{j \in C} c(j, Y) + \sum_{i \in Y} v(i),$$

where $c(j, Y) = \min_{i \in Y} c(j, i)$. The incremental problem is to compute an ordered sequence of the facilities F, $(f_1, f_2, \cdots, f_n)$ and a *threshold sequence*[2] $t_1 \leq t_2 \leq \cdots \leq t_n$ drawn from $\mathcal{R} \cup \infty$, such that for any scaling factor $\lambda > 0$ with k being the smallest index such that $t_k \geq \lambda$, $\mathsf{cost}_\lambda(C, \{f_i | i \leq k\})$ is a good approximation to $\mathsf{Opt}_\lambda = \min_{Y \subseteq F} \mathsf{cost}_\lambda(C, Y)$.

To fit this problem into the framework described in Section 2.2, we conceptually reformulate the problem. For simplicity, we assume in the sequel that the opening cost for every location is positive. At the end of this section, we show how this assumption can be removed. Each solution element S of U is a subset of $F \times \mathcal{R}$ that satisfies the condition that for two distinct x and y in S, $\pi_1(x) \neq \pi_1(y)$, where π_i is the projection to the i^{th} coordinate, $i = 1, 2$. For nonempty $S \in U$, define $\mathsf{ben}(S) = \max\{\pi_2(s) : s \in S\}$. The binary operator is

[2]The definition of threshold sequence in [55] is slightly different from ours, but serves the same purpose.

defined as $S_1 \preceq S_2$ iff $S_1 \subseteq S_2$. If $S \neq \emptyset$, the cost function is defined as

$$\mathsf{cost}(S) = \mathsf{ben}(S) \cdot \sum_{j \in C} c(j, \pi_1(S)) + \sum_{i \in \pi_1(S)} v(i).$$

We define $\mathsf{cost}(\emptyset)$ and $\mathsf{ben}(\emptyset)$ to be both 0.

The incremental ordering $(S_1, \ldots, S_k)$ yields a sequence of facility-threshold pairs $\{(f_1, t_1), \ldots, (f_n, t_n)\}$, where S_i is a prefix of the sequence. Note that this form naturally gives the threshold sequence $(t_1, \ldots, t_n)$.

We now establish the augmentation property for incremental facility location. We also present a companion CostBound subroutine that is monotonic in the benefit parameter; this helps prove the polynomial-time complexity of INCAPPROX for incremental facility location in Theorem 9.

Lemma 8 *There exists a $(1,1)$-augmentation for the λ-facility location problem. A $(1,\alpha)$-augmentation can be computed efficiently, where $\alpha = 1.5$. Furthermore, a monotonic* CostBound *function can be defined.*

Proof: Given an $S \subseteq F \times \mathcal{R}$, and $\lambda > 0$, if $\mathsf{ben}(S) \geq \lambda$, we are done. Hence, we assume $\mathsf{ben}(S) < \lambda$. Let $F^* \subseteq F$ be an optimal solution for λ-facility location problem. (Equivalently, $S^* = F^* \times \lambda$.) We augment S to S' as follows.

1. Initialize: $S' \leftarrow S$.
2. If $|F^* \backslash \pi_1(S')| > 1$, goto Step 3. Otherwise, goto Step 4.
3. Pick one $f \in F^* \backslash \pi_1(S')$, append $(f, \mathsf{ben}(S))$ to S'. Goto Step 2.
4. For the $f \in F^* \backslash \pi_1(S')$, append (f, λ) to S'. Exit.

By construction, $\mathsf{ben}(S') = \lambda$. Since $\pi_1(S^*) \subseteq \pi_1(S')$, $c(j, \pi_1(S')) \leq c(j, \pi_1(S^*))$.

The cost associated with S' is bounded as follows.

$$\begin{aligned}
\mathsf{ben}(S') \cdot \sum_{j \in C} c(j, \pi_1(S')) + \sum_{i \in \pi_1(S')} v(i) &\leq \mathsf{ben}(S^*) \cdot \sum_{j \in C} c(j, \pi_1(S^*)) \\
&\quad + \sum_{i \in \pi_1(S^*)} v(i) + \sum_{i \in \pi_1(S)} v(i) \\
&\leq \mathsf{cost}(S^*) + \mathsf{cost}(S).
\end{aligned}$$

To complete the proof of the lemma's first assertion, we have $\mathsf{Augment}(S, \lambda, 1, 1)$ return S' and $\mathsf{CostBound}(S, \lambda, 1, 1)$ return $\mathsf{cost}(S) + \mathsf{cost}(S^*)$. By construction and the bound on the cost, the desired properties of the augmentation subroutine, as given in Definition 2, hold. Moreover, since the cost of an optimal solution for λ-facility location is nondecreasing with increasing λ, the CostBound subroutine is monotonic in the benefit parameter.

To prove the second assertion of the lemma, we replace F^* with the poly-time computable 1.5-approximate solution due to Byrka [16]. The subroutine $\mathsf{Augment}(S, \lambda, 1, 1.5)$ returns S' and $\mathsf{CostBound}(S, \lambda, 1, 1.5)$ returns $1.5\textsf{LP-Opt}_\lambda + \mathsf{cost}(S)$, where $\textsf{LP-Opt}_\lambda$ is the cost of an optimal fractional solution to the λ-facility location problem. By [16], the solution F^* computed has cost at most $1.5\textsf{LP-Opt}_\lambda$. By definition and the upper bound on cost, the desired properties of the augmentation subroutine hold. Furthermore, CostBound is monotonic since the cost of an optimal fractional solution to the λ-facility location problem is nondecreasing with increasing λ. $\square$

Theorem 9 *There exists an incremental solution for the incremental facility location problem with competitive ratio* 4. *Moreover, an incremental solution of ratio* 4α *and a randomized solution of expected ratio* $e\alpha$ *can be computed efficiently, where* $\alpha = 1.5$.

Proof: Since every opening cost is nonzero and hence at least one, the cost of any nonempty solution is at least the minimum opening cost, which is one. By definition, the cost of $\emptyset$ is zero. Therefore, the condition of Theorem 1 that the cost of a solution be either zero or at least one holds, and the desired claims about the incremental approximation ratio follow from Lemma 8 and Theorem 1.

It remains to argue that the incremental solutions of ratios 4α and $e\alpha$ can be computed efficiently. In particular, we need to argue that the search procedure during each iteration (including the initialization step) of IncApprox can be implemented in polynomial time. That is, we need to show how to determine the largest value λ for which $\mathsf{CostBound}(S, \lambda, \gamma, \delta)$ is at most a certain value C (zero in the initialization step and $\beta_0\beta^{i+1}$ in iteration i). The monotonicity of $\mathsf{CostBound}(S, \lambda, \gamma, \delta)$ with respect to λ suggests a binary search over λ. Since the range for λ is the set of positive reals, however, it is not obvious whether such a binary search will terminate in polynomial time. We argue that the desired λ can be obtained using binary search within an interval $[\lambda_0, \lambda_{max}]$ such that (a) the ratio λ_{max}/λ_0 is at most exponential in the size of the instance, and (b) the binary search can terminate when we have identified an interval whose size is inverse exponential in the size of the instance.

Consider any set Y of facilities. As λ increases, the cost of Y, as a solution for the λ-facility location problem, increases linearly in λ. So each possible subset Y is a line in a two-dimensional graph that plots the cost of Y, as a solution to λ-facility location, as a function of λ. The optimal solutions to the λ-facility location problem are then given by the lower envelope of these lines. Each vertex on this lower envelope represents a λ for which there exist $F_1, F_2 \subseteq F$ such that (a) $\mathsf{cost}_\lambda(C, F_1) = \mathsf{cost}_\lambda(C, F_2)$, and (b) the connection cost and opening cost of F_1 are different than those of F_2. The latter condition is true since F_1 and F_2

correspond to two different lines in the plot. We thus have

$$\lambda \cdot \sum_{j \in C} c(j, F_1) + \sum_{i \in F_1} v(i) = \lambda \cdot \sum_{j \in C} c(j, F_2) + \sum_{i \in F_2} v(i),$$

where $\sum_{j \in C} c(j, F_1) \neq \sum_{j \in C} c(j, F_2)$ and $\sum_{i \in F_1} v(i) \neq \sum_{i \in F_2} v(i)$. Suppose c_{max} and v_{max} are the largest distance and opening cost, respectively. Since all the distances and facility opening costs are integers, we obtain that λ is a rational number whose denominator is at most nc_{max} and the numerator at most nv_{max}. Thus, λ_{max} is at most nv_{max}, and λ_0 and the smallest difference between the λ values corresponding to two adjacent vertices of the lower envelope are at least $1/(nc_{max})$. Consequently a binary search over λ within the range $[1/(nc_{max}), nv_{max}]$ will terminate in time $\log(n^2 v_{max}(c_{max})^2)$, which is polynomial in the size of the instance. $\square$

Finally, we consider the case where the opening cost of a facility could be zero. In this case, the condition of Theorem 1 that the cost of a solution be either zero or strictly bounded away from zero may not hold. To see this, let Y_0 be the subset of facilities in F where the opening cost is zero. Then, the cost of the solution $Y_0 \times \{\lambda\}$ tends to zero as λ tends to zero.

We now claim that the solution $S_\lambda = Y_0 \times \{\lambda\}$ achieves optimum cost for $\lambda \leq \lambda_0 = 1/(nc_{max})$, where c_{max} is the maximum distance between any two points. To see this, we first note that for $\lambda \leq 1/(nc_{max})$, the cost of any facility location solution $X \subset Y_0$ is at least that of Y_0 since the opening costs are zero for both solutions while the connecting cost of Y_0 is at most that of X. On the other hand, any solution that opens a facility outside Y_0 has a cost of at least 1, while the cost of Y_0 is at most $\lambda nc_{max} \leq 1$.

We next redefine the set of feasible solutions as follows. Each solution element S of U is a subset of $F \times \mathcal{R}$ such that for any two distinct x and y in S, $\pi_1(x) \neq \pi_1(y)$, and for all $f \in Y_0$, (f, λ_0) is in S. We now apply the framework of Section 2.2 to the incremental facility location and run INCAPPROX with initial solution $S_0 = Y_0 \times \{\lambda_0\}$ instead of $\emptyset$. In this case, the cost of any solution is either zero or strictly bounded away from zero. So we apply Lemma 8 and Theorem 1 to compute an ordered sequence of the facilities $(f_1, f_2, \ldots, f_n)$ and a threshold sequence such that for any scaling factor $\lambda > \lambda_0$, $\mathsf{cost}_\lambda(C, \{f_i | i \leq k\})$ is a good approximation to $\mathsf{Opt}_\lambda = \min_{Y \subseteq F} \mathsf{cost}_\lambda(C, Y)$, where k is the smallest index such that $t_k \geq \lambda$. Note that the first $|Y_0|$ facilities in this sequence are the elements of Y_0 while the first $|Y_0|$ thresholds are all λ_0. To obtain an incremental solution for all $\lambda > 0$, we return the same ordered sequence of facilities and the threshold sequence to be the same as before except that the first $|Y_0|$ values are all zeros instead of λ_0. For any $\lambda \in (0, \lambda_0]$, the incremental facility location solution returned is Y_0, which is optimal. For $\lambda > \lambda_0$, the desired competitive ratio(s) hold.

2.3.4 Incremental covering problems

The incremental k-set cover problem

Given a universe X of n elements and a collection of subsets of the universe, $\mathcal{C} = \{C_1, \cdots, C_m\}$ and a cost function $c : \mathcal{C} \rightarrow Q^+$, find an ordered sequence of $\mathcal{C}$, such that for *any* $k \in [1, n]$, the minimal prefix of the sequence that covers k elements is a good approximation to the k-set cover problem. Recall that the k-set cover problem asks for a min-cost subcollection of $\mathcal{C}$ that covers at least k elements.

In the language of Section 2.2, the universe U is $2^{\mathcal{C}}$. The benefit of $S \subseteq \mathcal{C}$ is simply the total number of elements covered by S. Then $S_1 \preceq S_2$ iff $S_1 \subseteq S_2$, and $\mathsf{cost}(S)$ is the sum of the weights of the subsets in S.

Lemma 9 *There exists a $(1,1)$-augmentation for k-set cover problem. Moreover, a $(1,\alpha)$-augmentation can be computed efficiently, where $\alpha = \ln n + 1$.*

Proof: The proof is straightforward. Given a partial solution S and k, if $\mathsf{ben}(S) \geq k$, we are done. Otherwise, let $S(k)$ be any set collection with $\mathsf{ben}(S(k)) \geq k$. Clearly, by setting $S' = S \cup S(k)$, we have $\mathsf{ben}(S') \geq k$ and its cost is bounded by

$$\mathsf{cost}(S) + \mathsf{cost}(S(k)).$$

Using the optimum solution to k-set cover for $S(k)$ proves the first assertion. And using a poly-time computable α-approximate solution (e.g., [58] gives $\alpha = \ln n + 1$) for $S(k)$ proves the second assertion. $\square$

Theorem 10 *There exists a solution for the incremental k-set cover problem with competitive ratio 4. Moreover, a solution with ratio 4α can be computed efficiently, where $\alpha = \ln n + 1$.*

Proof: Follows from Lemma 9 and Theorem 1. $\square$

The incremental k-vertex cover problem

Just as vertex cover is a special case of set cover, k-vertex cover problem is a special case of k-set cover, where each edge (resp., vertex) in the vertex cover problem is an element (resp., set) in the set cover poblem. We hence have a corresponding incremental vertex cover problem. A 2-approximation algorithm for k-vertex cover is known [15, 30, 47].

Theorem 11 *There exists an incremental solution for the incremental k-vertex cover problem with competitive ratio 4. Moreover, a solution with ratio 4α can be computed efficiently, where $\alpha = 2$.*

2.4 A general approach for problems with bounded envelope

Consider a problem Π specified by a quadruple $\langle U, \preceq, \mathsf{ben}, \mathsf{cost}\rangle$ as discussed in Section 2.2. We additionally assume that the range of benefit function is positive integers with maximum value B_{max}.

Definition 5 *An α-approximate bounded envelope of a problem Π consists of values b_k for k ranging from 1 to B_{max} and solutions S_{n_i} with $\mathsf{ben}(S_{n_i}) = n_i$ for $n_i \in \mathcal{E} = \{n_1, n_2, \ldots, n_l\}$ and $1 = n_1 < n_2 < \cdots < n_l = B_{max}$, such that:*

1. $b_k \leq \mathsf{cost}(\mathsf{Opt}(k))$ *for* $1 \leq k \leq B_{max}$*;*
2. $\mathsf{cost}(S_{n_i}) \leq \alpha \cdot b_{n_i}$ *for* $1 \leq i \leq l$*;*
3. $b_k = b_{n_{i-1}} + \frac{k-n_{i-1}}{n_i-n_{i-1}}(b_{n_i} - b_{n_{i-1}})$ *for* $n_{i-1} \leq k \leq n_i$ *and* $i = 2, \ldots, l$*.*

In this section we give a slightly modified version of the general algorithm from Section 2.2 for incremental versions of problems which have a bounded envelope with some factor that is better than the approximation factor from the approximation algorithm for that problem. This algorithm allows us to replace the approximation algorithm factors from the algorithms in Section 2.2

with the factor from the bounded envelope. In particular though a $(3+\epsilon)$ is the best known performance guarantee for the k-median problem, there is a 2-approximate bounded envelope for the problem which allows us to replace the factor of $(3+\epsilon)$ by 2 thereby significantly improving the performance guarantee.

The idea of an α-approximate bounded envelope was first used for the k-MST and the minimum latency problems by Archer, Levin, and Williamson [5, 4].

Definition 6 *An interpolation algorithm $\mathcal{I}$ for a problem Π is defined as an algorithm which when given two solutions S_1 and S_2 with $S_1 \preceq S_2$ and $\mathsf{ben}(S_1) < \mathsf{ben}(S_2)$, outputs a solution S such that $S_1 \preceq S \preceq S_2$, $\mathsf{ben}(S) = \mathsf{ben}(S_1) + 1$, and $\mathsf{cost}(S) \leq \mathsf{cost}(S_1) + \frac{1}{\mathsf{ben}(S_2)-\mathsf{ben}(S_1)}(\mathsf{cost}(S_2) - \mathsf{cost}(S_1))$.*

We use an idea similar to the generalized approach for incremental algorithms in Section 2.2 to get a better incremental algorithm for the problem Π given an α-approximate bounded envelope, interpolation algorithm and an augmentation algorithm $\mathsf{Augment}(S, p, \gamma, \delta)$ for that problem.

Here we assume that the subroutine $\mathsf{Augment}(S, p, \gamma, \delta)$ for $p \in \mathcal{E}$ finds a solution S' with benefit at least p and $S \prec S'$ such that

$$\mathsf{cost}(S') \leq \gamma \mathsf{cost}(S) + \delta b_p \tag{2.2}$$

which is slightly stronger than Definition 2. This is accomplished by using solutions from the bounded envelope in the (γ, δ)-*Augmentation* as in Definition 2. We also make sure that our algorithm calls the augmentation procedure only for benefit $p \in \mathcal{E}$.

Now we present an algorithm for the incremental version of a problem Π using an α-approximate bounded envelope, an augmentation procedure and an interpolation algorithm $\mathcal{I}$ for the problem Π.

Algorithm 3: BOUNDEDINCAPPROX(γ, δ)

1. Initialization:

   If $\mathsf{cost}(\emptyset) = 0$ then $S_0 = \mathsf{Augment}(\emptyset, q, \gamma, \delta)$, where q is the largest value for which $\mathsf{CostBound}(\emptyset, q, \gamma, \delta) = 0$. $S_0 = \emptyset$ otherwise.

   1D: (Deterministic) $i = 0$, $\beta = 2\gamma$, $\beta_0 = \beta$.

   1R: (Randomized) $i = 0$, β is the minimizer of $\frac{\beta-1}{(1-\gamma/\beta)\ln\beta}$, $\beta_0 = \beta^X$, where X is uniform from $[0, 1)$.

2. Iteration i: $S_{i+1} = \mathsf{Augment}(S_i, p, \gamma, \delta)$, where p is the largest value in the set $\mathcal{E} = \{n_1, n_2, \ldots, n_l\}$ from the bounded envelope for which $\mathsf{CostBound}(S_i, p, \gamma, \delta)$ is at most $\beta_0\beta^{i+1}$.

3. Termination: If $\mathsf{ben}(S_{i+1}) \neq B_{max}$ then $i \leftarrow i + 1$, go to step 2. Otherwise, return sequence $S_1, \cdots, S_{i+1}$ along with the solutions for all the intermediate benefit values obtained by interpolating between S_i's repeatedly using the interpolation algorithm.

Theorem 12 *If (γ, δ)-augmentation holds for reals $\gamma \geq 1$, $\delta > 0$, then each S_j that* BOUNDEDINCAPPROX(γ, δ) *outputs is a (i) $4\gamma\delta$-approximate solution (Deterministic); (ii) $\min_\beta \frac{\delta(\mu-1)}{(1-\gamma/\mu)\ln\mu}$-approximate solution (Randomized) which equals $e\delta$, when $\gamma = 1$.*

Proof: Here the proof is similar to the proof of Theorem 1. Here we assume that costs can be scaled such that for any solution $S \in U$, $\mathsf{cost}(S)$ is either zero or $\geq \beta$.

Note that we maintain the invariant

$$\mathsf{cost}(S_i) \leq \beta_0 \beta^i \tag{2.3}$$

in Step (2) of the algorithm for all $i \geq 0$.

Consider any $n_j \in \mathcal{E}$. Let k be the smallest integer such that

$$\delta b_{n_j} \leq \beta_0 \beta^k \cdot (1 - \gamma/\beta). \tag{2.4}$$

It follows that

$$b_{n_j} > \beta_0 \beta^{k-1} \cdot \frac{1 - \gamma/\beta}{\delta}. \tag{2.5}$$

$$\begin{aligned}
\mathsf{cost}(\mathsf{Augment}(S_{k-1}, n_j, \gamma, \delta)) &\leq \gamma \mathsf{cost}(S_{k-1}) + \delta b_{n_j} \qquad (\textit{By Equation } 2.2) \\
&\leq \gamma \beta_0 \beta^{k-1} + \beta_0 \beta^k (1 - \gamma/\beta) \qquad (2.6) \\
&\qquad (\textit{By Equations } 2.3 \textit{ and } 2.4) \\
&\leq \beta_0 \beta^k. \qquad (2.7)
\end{aligned}$$

Since by Step (2), $S_k = \mathsf{Augment}(S_{k-1}, p, \gamma, \delta)$ for the largest value of p in $\mathcal{E}$ such that $\mathsf{cost}(S_k) \leq \beta_0 \beta^k$, Equation 2.7 implies $n_j \leq \mathsf{ben}(S_k)$. Let r be such that $\mathsf{ben}(S_{r-1}) < n_j \leq \mathsf{ben}(S_r)$. Then $n_j \leq \mathsf{ben}(S_r) \leq \mathsf{ben}(S_k)$.

Deterministic case: The approximation factor for the solution obtained with benefit n_j is no more than

$$\begin{aligned}
\mathsf{cost}(S_r)/b_{n_j} &\leq \mathsf{cost}(S_k)/b_{n_j} \\
&\leq \frac{\beta_0 \beta^k}{\beta_0 \beta^{k-1} \cdot \frac{1-\gamma/\beta}{\delta}} = \beta^2 \delta / (\beta - \gamma).
\end{aligned}$$

This approximation factor holds for all the solutions returned by the algorithm with benefit $n_i \in \mathcal{E}$. Consider any solution S with benefit k returned by the algorithm with $n_{j-1} < k \leq n_j$. Let the solutions returned by the algorithm with benefits n_{j-1} and n_j be S_1 and S_2 respectively.

$$\begin{aligned}
\mathsf{cost}(S) &\leq \mathsf{cost}(S_1) + \frac{k - n_{j-1}}{n_j - n_{j-1}} \left(\mathsf{cost}(S_2) - \mathsf{cost}(S_1)\right) \\
&\leq \frac{\beta^2\delta}{\beta - \gamma} \left(b_{n_{j-1}} + \frac{k - n_{j-1}}{n_j - n_{j-1}} (b_{n_j} - b_{n_{j-1}}) \right) \\
&= \frac{\beta^2\delta}{\beta - \gamma} b_k \\
&\leq \frac{\beta^2\delta}{\beta - \gamma} \mathsf{cost}(\mathsf{Opt}(k))
\end{aligned}$$

The first inequality follows from the property of interpolation algorithm and the last equality follows from the property of the lower bounds of the bounded envelope.

The above bound is minimized when $\beta = 2\gamma$, thus yielding a $4\delta\gamma$-approximation factor.

Randomized case: The proof is very similar to the proof of the randomized case of Theorem 1. □

As in the case of INCAPPROX, the running time of BOUNDEDINCAPPROX is dominated by the calls to the augmentation subroutine. We can define Cost-Bound to simply return the cost of the augmented solution, and perform a linear search to find the largest p in the set $\{n_1, n_2, \ldots, n_l\}$ with the desired property in each iteration. Then the number of calls per iteration is $O(B_{max})$, and since the number of iterations is $O(\log_\beta \mathsf{Maxcost})$, the number of calls to the augmentation subroutine is bounded by $O(B_{max} \cdot \log_\beta \mathsf{Maxcost})$.

We can obtain an improved running time if we define a CostBound subroutine that is monotonic in the benefit parameter and then perform a binary search to find the largest p in the set $\{n_1, n_2, \ldots, n_l\}$ with the desired property in each iteration. In this case, the number of calls per iteration is $O(\log B_{max})$, leading to a bound of $O(\log_\beta \mathsf{Maxcost} \cdot B_{max})$ on the number of calls to the augmentation subroutine.

2.5 Applications to incremental and hierarchical median problems

In this section, we apply our framework of Section 2.4 to incremental and hierarchical median problems.

Theorem 13 *The procedure* $\mathsf{Augment}(S, p, 1, 2\alpha)$ *for* $p \in \mathcal{E}$ *can be efficiently implemented for the incremental median problem when an* α*-approximate bounded envelope with breakpoint set* $\mathcal{E}$ *is given.*

Proof: Using the α-approximate solution with benefit p from the bounded envelope for S_1 in the proof of Lemma 2 gives us the required result. Note that here $\mathsf{cost}(\mathsf{Augment}(S, p, 1, 2\alpha)) \leq \mathsf{cost}(S) + 2\alpha b_p$ where b_p is the lower bound for benefit p provided by the bounded envelope as required by the algorithm BOUNDEDINCAPPROX(γ, δ). To define a monontonic CostBound subroutine, we have $\mathsf{CostBound}(S, p, 1, 2\alpha)$ return $\mathsf{cost}(S) + 2\alpha b_p$. □

Theorem 14 *An interpolation algorithm exists for the incremental median problem.*

Proof: Here we provide the procedure to find a k-median solution S when two solutions S_1 and S_2 with $S_1 \prec S_2$ and $\mathsf{ben}(S_1) < \mathsf{ben}(S_2)$ are given such that $\mathsf{ben}(S) = \mathsf{ben}(S_1) + 1$ and $\mathsf{cost}(S) \leq \mathsf{cost}(S_1) + \frac{1}{\mathsf{ben}(S_2) - \mathsf{ben}(S_1)}(\mathsf{cost}(S_2) - \mathsf{cost}(S_1))$.

Since $S_1 \prec S_2$, $S_2 \subset S_1$. Let the assignment corresponding to assigning the clients to the closest facility in S_1 and S_2 be a_1 and a_2. For every facility $h \in S_1 \backslash S_2$ define $\Delta(h) = \sum_{j:a_1(j)=h}(c_{j,a_2(j)} - c_{j,a_1(j)})$. Then $\Delta(h)$ is the change in cost by shifting all the clients assigned to h by a_1 to the facilities they are assigned to by a_2. Observe that $\mathsf{cost}(S_1, a_1) - \mathsf{cost}(S_2, a_2) = \sum_{h \in S_1 \backslash S_2} \Delta(h)$ since assignment a_1 and a_2 assign clients to the nearest facility in S_1 and S_2 respectively.

Consider the facility $i \in S_1 \setminus S_2$ with minimum Δ value. Define an assignment a such that $a(j) = a_2(j)$ if $a_1(j) = i$ and $a(j) = a_1(j)$ otherwise. Let $S = S_1 \setminus \{i\}$.

$$
\begin{aligned}
\mathsf{cost}(S) - \mathsf{cost}(S_1) &\leq \mathsf{cost}(S, a) - \mathsf{cost}(S_1, a_1) \\
&= \Delta(i) \\
&\leq \frac{1}{\mathsf{ben}(S_2) - \mathsf{ben}(S_1)} \sum_{h \in S_1 \backslash S_2} \Delta(h) \\
&= \frac{1}{\mathsf{ben}(S_2) - \mathsf{ben}(S_1)} (\mathsf{cost}(S_2) - \mathsf{cost}(S_1))
\end{aligned}
$$

So we have found S such that $\mathsf{ben}(S) = \mathsf{ben}(S_1) + 1$, $S_1 \prec S \prec S_2$ and $\mathsf{cost}(S) \leq \mathsf{cost}(S_1) + \frac{1}{\mathsf{ben}(S_2) - \mathsf{ben}(S_1)}(\mathsf{cost}(S_2) - \mathsf{cost}(S_1))$.

$\square$

Theorem 15 *The procedure* $\mathsf{Augment}(S, p, 3, 2\alpha)$ *for* $p \in \mathcal{E}$ *can be efficiently implemented for the hierarchical median problem when an* α*-approximate bounded envelope* *with breakpoint set $\mathcal{E}$ is given.*

Proof: Using the α-approximate solution with benefit p from the bounded envelope for S_1 in the proof of Lemma 4 gives us the required result. Note that here $\mathsf{cost}(\mathsf{Augment}(S, p, 3, 2\alpha)) \leq 3\mathsf{cost}(S) + 2\alpha b_p$ where b_p is the lower bound for benefit p provided by the bounded envelope as required by the algorithm in Section 2.4. $\square$

Theorem 16 *An interpolation algorithm exists for the hierarchical median problem.*

Proof: The interpolation procedure for the hierarchical median problem is exactly same as in the proof of Theorem 14. Here again $\mathsf{cost}(S_1, a_1) - \mathsf{cost}(S_2, a_2) = \sum_{h \in S_1 \setminus S_2} \Delta(h)$ since $(S_1, a_1) \prec (S_2, a_2)$ and so the assignment of clients differ only for the clients assigned to the facilities in $S_1 \setminus S_2$ in a_1.

This gives (S, a) such that $(S_1, a_1) \prec (S, a) \prec (S_2, a_2)$, $\mathsf{ben}(S, a) = \mathsf{ben}(S_1, a_1) + 1$ and $\mathsf{cost}(S, a) \leq \mathsf{cost}(S_1, a_1) + \frac{1}{\mathsf{ben}(S_2) - \mathsf{ben}(S_1)}(\mathsf{cost}(S_2, a_2) - \mathsf{cost}(S_1, a_1))$. $\square$

Theorem 17 *A 2-approximate bounded envelope for the incremental and hierarchical median problem can be efficiently computed.*

Proof: We prove the existence of a 2-approximate bounded envelope in two stages. First we define a Lagrangean Multiplier Preserving (LMP) Facility Location (FL) algorithm and show the existence of a $(2 - 1/n)$-approximate LMP FL algorithm in Lemma 10. Then we show that any $(\alpha - \epsilon)$-approximate LMP FL algorithm can be used to get an α-approximate bounded envelope for the k-median problem in Lemma 13. This bounded envelope can be used in the case of both incremental and hierarchical median problem. This proves the theorem.

□

2.5.1 The metric facility location problem and LMP facility location algorithm

Here we consider the standard metric *facility location problem*. As in the case of the k-median problem, we have F a set of facilities and C a set of clients in a metric space. Let f_i be the opening cost of facility i and c_{ij} be the cost of connecting client j to facility i; c_{ij} will be the distance between client j and facility i. The goal of the facility location problem is to find a subset of facilities to open so that the cost of opening these facilities and connecting each client to an open facility is minimized. Let $n_c = |C|$, $n_f = |F|$, $n_c + n_f = n$ and $n_c * n_f = m$. In this section we use $\mathsf{cost}(S)$ and $c(S)$ interchangeably.

Here we consider the LP relaxation of standard integer programming formulation of the problem. The 0-1 variable y_i denotes whether the facility i is open and x_{ij} is a integer 0-1 variable denoting whether the client j is connected to facility i.

$$\begin{array}{llll}
 & \text{Min} & \sum_{i\in F} f_i y_i + \sum_{i\in F, j\in C} c_{ij}x_{ij} & \\
 & \text{subject to:} & & \\
 & & \sum_{i\in F} x_{ij} = 1 & \forall j \in C \\
(FL-P) & & x_{ij} \le y_i & \forall i \in F, \forall j \in C \\
 & & x_{ij} \ge 0 & \forall i \in F, \forall j \in C \\
 & & y_i \ge 0 & \forall i \in F.
\end{array}$$

The dual of this LP relaxation is

$$\begin{array}{llll}
 & \text{Max} & \sum_{j\in C} v_j & \\
 & \text{subject to:} & & \\
 & & \sum_{j\in C} w_{ij} \le f_i & \forall i \in F \\
(FL-D) & & v_j - w_{ij} \le c_{ij} & \forall i \in F, \forall j \in C \\
 & & w_{ij} \ge 0 & \forall i \in F, \forall j \in C.
\end{array}$$

Definition 7 *An α-approximation algorithm $\mathcal{A}$ is said to be α-*Lagrangean multiplier preserving *(α-LMP) if it outputs an integral solution $(\hat{x}, \hat{y})$ feasible for $(FL-P)$ and a dual solution $(\hat{v}, \hat{w})$ feasible for $(FL-D)$ such that*

$$\sum_{i\in F, j\in C} c_{ij}\hat{x}_{ij} + \alpha \sum_{i\in F} f_i \hat{y}_i \le \alpha \sum_{j\in C} \hat{v}_j. \tag{2.8}$$

Lemma 10 *There exists a $(2-1/n)$-approximate Lagrangean Multiplier Preserving Facility Location algorithm.*

Proof: We want to find an integer solution $(\bar{x}, \bar{y})$ to the facility location problem and a dual feasible solution to the facility location dual such that

$$\sum_{i \in F, j \in D} c_{ij}\bar{x}_{ij} + \beta \sum_{i \in F} f_i \bar{y}_i \leq \beta \sum_{j \in D} v_j$$

for $\beta = 2 - \frac{1}{n}$.

We run the Jain et al.'s second greedy algorithm [41] on the problem instance with the same service costs, but facility costs $\hat{f}_i = \beta f_i$. Now by the theorem (Theorem 8.3) in their paper, for any given set S of clients, they have that $\sum_{j \in S} \alpha_j - \hat{f}_i \leq \beta \sum_{j \in S} c_{ij}$. Furthermore, the greedy algorithm gives an integer solution $(\bar{x}, \bar{y})$ such that

$$\sum_{j \in D} \alpha_j = \sum_{i \in F, j \in D} c_{ij}\bar{x}_{ij} + \sum_{i \in F} \hat{f}_i \bar{y}_i = \sum_{i \in F, j \in D} c_{ij}\bar{x}_{ij} + \beta \sum_{i \in F} f_i \bar{y}_i.$$

To get our desired result, we set $v_j = \alpha_j / \beta$. Then we have immediately that

$$\sum_{i \in F, j \in D} c_{ij}\bar{x}_{ij} + \beta \sum_{i \in F} f_i \bar{y}_i \leq \beta \sum_{j} v_j.$$

Furthermore, we claim v is a feasible solution to the facility location dual. To see this, pick an arbitrary $i \in F$, and let $S \subseteq F$ be the facilities such that $v_j - c_{ij} \geq 0$. Then by the Jain et al. result, we have that

$$\sum_{j \in S} \alpha_j - \hat{f}_i \leq \beta \sum_{j \in S} c_{ij},$$

or

$$\beta \sum_{j \in S} v_j - \beta f_i \leq \beta \sum_{j \in S} c_{ij},$$

or

$$\sum_{j \in S} (v_j - c_{ij}) \leq f_i,$$

or

$$\sum_{j \in D} \max(v_j - c_{ij}, 0) \leq f_i$$

by the choice of S.

By taking $w_{ij} = \max(v_j - c_{ij}, 0)$, (v, w) is a feasible solution to the facility location dual $(FL - D)$. $\square$

2.5.2 Getting a bounded envelope for k-median problem

Here we give an algorithm which gives an $(\alpha + \epsilon)$-bounded envelope for the k-median problem given an α-LMP algorithm for metric uncapacitated facility location problem for any $\epsilon > 0$. In essence given an α-LMP algorithm for facility location problem, we arrive at k-median solutions $S_{n_1}, \ldots, S_{n_l}$ for $1 = n_1 < \cdots < n_l = n$ with $|S_k| = k$ and $b_1, \ldots, b_n$ satisfying the following properties. Let the cost of optimal k-median solution be denoted by OPT_k. Let us fix an $\epsilon > 0$.

1. $b_k \leq OPT_k$ for $1 \leq k \leq n$
2. $c(S_{n_i}) \leq (\alpha + \epsilon) \cdot b_{n_i}$ for $1 \leq i \leq l$
3. $b_k = b_{n_{i-1}} + \frac{b_{n_i} - b_{n_{i-1}}}{n_i - n_{i-1}}(k - n_{i-1})$ for $n_{i-1} \leq k \leq n_i$ and $i = 2, \ldots, l$

Let us consider the following integer program for the k-median problem:

$$\text{Min} \quad \sum_{i \in F, j \in C} c_{ij} x_{ij}$$

subject to:

$$\begin{aligned}
&\sum_{i \in F} x_{ij} = 1 && \forall j \in C \\
&x_{ij} \leq y_i && \forall i \in F, \forall j \in C \\
&\sum_{i \in F} y_i \leq k && \\
&x_{ij} \in \{0, 1\} && \forall i \in F, \forall j \in C \\
&y_i \in \{0, 1\} && \forall i \in F.
\end{aligned}$$

The LP relaxation of the program is

$$(k-P) \qquad \begin{array}{rll} \text{Min} & \displaystyle\sum_{i\in F, j\in C} c_{ij}x_{ij} & \\ \text{subject to:} & & \\ & \displaystyle\sum_{i\in F} x_{ij} = 1 & \forall j \in C \\ & x_{ij} \leq y_i & \forall i \in F, \forall j \in C \\ & \displaystyle\sum_{i\in F} y_i \leq k & \\ & x_{ij} \geq 0 & \forall i \in F, \forall j \in C \\ & y_i \geq 0 & \forall i \in F. \end{array}$$

The dual of the LP relaxation is

$$(k-D) \qquad \begin{array}{rll} \text{Max} & \displaystyle\sum_{j\in C} v_j - zk & \\ \text{subject to:} & & \\ & \displaystyle\sum_{j\in C} w_{ij} \leq z & \forall i \in F \\ & v_j - w_{ij} \leq c_{ij} & \forall i \in F, \forall j \in C \\ & w_{ij} \geq 0 & \forall i \in F, \forall j \in C \\ & z \geq 0. & \end{array}$$

The essential idea is as follows: to obtain the k-median solutions we use an α-LMP algorithm for the facility location problem. Let us call this algorithm $\mathcal{A}$. We parameterize this algorithm with a common facility opening cost z for all facilities. This algorithm outputs integral primal solution $(\hat{x}, \hat{y})$ and dual solution $(\hat{v}, \hat{w})$ from which we show that we can arrive at a solution S and a lower bound $\hat{b}$ such that $c(S) \leq \alpha\hat{b}$ and $\hat{b} \leq OPT_{|S|}$. We do a binary search on z to obtain a solution with exactly k nodes for each k. If we find such a solution, we add the solution and the lower bound to our collection. If not, we do a binary search on z until we get sufficiently close z_1 and z_2 and two solutions S_1 and S_2 with their corresponding lower bounds $\hat{b}_1$ and $\hat{b}_2$ such that $|S_1| < k$ and $|S_2| > k$. We show that if z_1 and z_2 are close enough then an interpolation of $\hat{b}_1$ and $\hat{b}_2$ after a slight modification gives a lower bound on the optimal solution with k nodes. Later we pick a subset of solutions from the collection in order to obtain the desired properties.

Since $\mathcal{A}$ is an α-LMP algorithm it outputs a integral solution $(\hat{x}, \hat{y})$ feasible for $(FL - P)$ and a feasible dual solution $(\hat{v}, \hat{w})$ feasible for $(FL - D)$ such that

$$\sum_{i\in F, j\in C} c_{ij}\hat{x}_{ij} + \alpha \sum_{i\in F} f_i\hat{y}_i \leq \alpha \sum_{j\in C} \hat{v}_j. \tag{2.9}$$

Theorem 18 *Let $\mathcal{A}$ return $(\hat{x}, \hat{y})$ and $(\hat{v}, \hat{w})$ when $f_i = z$ for all i. Suppose $\sum_{i\in F} \hat{y}_i = k$. Let $S_k = \{i : y_i = 1, i \in F\}$ and $\hat{b}_k = \sum_{j\in C} v_j - zk$. Then $c(S_k) \leq \alpha \cdot \hat{b}_k$ and $\hat{b}_k \leq OPT_k$.*

Proof: Since $f_i = z$ for all i, $(\hat{x}, \hat{y})$ is an integral solution feasible for $(FL - P)$ and $(\hat{v}, \hat{w})$ is a dual solution feasible for $(FL - D)$, we have an integral solution $(\hat{x}, \hat{y})$ feasible for $(k - P)$ and a dual solution $(\hat{v}, \hat{w}, z)$ feasible for $(k - D)$. Substituting $f_i = z$ for all i and $\sum_{i\in F} \hat{y}_i = k$ in (2.9) we get

$$\sum_{i\in F, j\in C} c_{ij}\hat{x}_{ij} + \alpha z k \leq \alpha \sum_{j\in C} \hat{v}_j \tag{2.10}$$

Using this we get the following bound on the cost of the solution S_k:

$$c(S_k) = \sum_{i\in F, j\in C} c_{ij}\hat{x}_{ij} \leq \alpha \left(\sum_{j\in C} \hat{v}_j - zk \right) = \alpha \hat{b}_k.$$

Since $(\sum_{j\in C}\hat{v}_j - zk)$ is the dual objective of $(k-D)$, by weak duality $\hat{b}_k = (\sum_{j\in C}\hat{v}_j - zk) \le OPT_k$. □

Let $c_{\max}$ be the maximum distance between a client and a facility. Here we assume that when $z = 0$ algorithm $\mathcal{A}$ outputs all facilities (that is, $y_i = 1\ \forall i \in F$), and when z is very large the algorithm $\mathcal{A}$ outputs only one facility. In particular, when $z = 2n^2 c_{max}$, the $(2 - \frac{1}{n_f})$-approximate LMP facility location algorithm by Jain et al. [41] opens just one facility. For each k, we do a binary search on the values of z to find the value of z for which the algorithm returns a solution with exactly k facilities open. If we succeed we return the solution S_k along with its lower bound $\hat{b}_k$. Otherwise we perform the binary search until $z_1 - z_2 \le \frac{\epsilon}{\alpha n_f} c_{\min}$ where $c_{\min}$ is the minimum nonzero distance between a facility and a client.

Here we scale down the lower bounds $\hat{b}_k$ before returning them by a factor of $\frac{\alpha}{\alpha+\epsilon}$ for technical reasons. Note that if ϵ is small the lower bounds are not decreased by too much. Let $b_k = \left(\frac{\alpha}{\alpha+\epsilon}\right)\hat{b}_k$.

Lemma 11 *Consider any solution S_k returned by the algorithm with its corresponding lower bound $b_k = \left(\frac{\alpha}{\alpha+\epsilon}\right)\hat{b}_k$. Then $c(S_k) \le (\alpha+\epsilon)\cdot b_k$.*

Proof:

$$c(S_k) \le \alpha \cdot \hat{b}_k = \alpha \cdot \left(\frac{\alpha+\epsilon}{\alpha}\right) b_k = (\alpha+\epsilon)\cdot b_k,$$

where the first inequality is by Theorem 18. □

Lemma 12 *Assume $OPT_k \ge c_{\min}$. Let S_1 and S_2 be the solutions returned by the $\mathcal{A}$ with facility opening costs set to z_1 and z_2 respectively and let b_1 and b_2 be the corresponding lower bounds. Let $k_1 = |S_1| < k$, $k_2 = |S_2| > k$, $z_1 - z_2 \le \frac{\epsilon}{\alpha n_f} c_{\min}$. Let $k = pk_1 + qk_2$ and $p + q = 1$ and $p, q \ge 0$. If we set $b_k = pb_1 + qb_2$ then $b_k \le OPT_k$.*

Proof: Let $(\hat{v}^1, \hat{w}^1)$ and $(\hat{v}^2, \hat{w}^2)$ be the dual solutions returned by the algorithm $\mathcal{A}$ corresponding to the facility opening costs z_1 and z_2. Let $v = p\hat{v}^1 + q\hat{v}^2$ and $w = p\hat{w}^1 + q\hat{w}^2$. Then

$$\begin{aligned}
b_k &= pb_1 + qb_2 \\
&= \frac{\alpha}{\alpha+\epsilon}\left(p\hat{b}_1 + q\hat{b}_2\right) \\
&= \frac{\alpha}{\alpha+\epsilon}\left(p\left(\sum_{j\in C} v_j^1 - z_1k_1\right) + q\left(\sum_{j\in C} v_j^2 - z_2k_2\right)\right) \\
&= \frac{\alpha}{\alpha+\epsilon}\left(\sum_{j\in C} v_j - pz_1k_1 - qz_1k_2 + qz_1k_2 - qz_2k_2\right) \\
&= \frac{\alpha}{\alpha+\epsilon}\left(\sum_{j\in C} v_j - z_1\left(pk_1+qk_2\right) + qk_2\left(z_1-z_2\right)\right) \\
&\le \frac{\alpha}{\alpha+\epsilon}\left(\sum_{j\in C} v_j - z_1k\right) + \frac{\alpha}{\alpha+\epsilon}\cdot qk_2\cdot\frac{\epsilon}{\alpha n_f}c_{min} \\
&\le \frac{\alpha}{\alpha+\epsilon}\left(\sum_{j\in C} v_j - z_1k\right) + \frac{\epsilon}{\alpha+\epsilon}c_{min} \\
&\le OPT_k
\end{aligned}$$

The last inequality holds since (v, w, z_1) is feasible for the dual $(k-D)$ by convexity of feasible region and since both $\left(\sum_{j\in C} v_j - z_1k\right)$ and $c_{\min}$ are lower bounds on OPT_k. $\square$

The lemma assumes that $OPT_k \ge c_{\min}$, which is false only if $OPT_k = 0$. To deal with this case, we run any approximation algorithm for the k-median problem for all values of k; note that if $OPT_k = 0$, an approximation algorithm will return an optimal solution of cost 0. For the values of k for which the algorithm returns a solution of cost 0, we set $b_k = 0$.

At the end we have solutions S_k for some specific values of k and their corresponding lower bounds b_k such that $c(S_k) \le (\alpha+\epsilon)\cdot b_k$. We plot these solutions

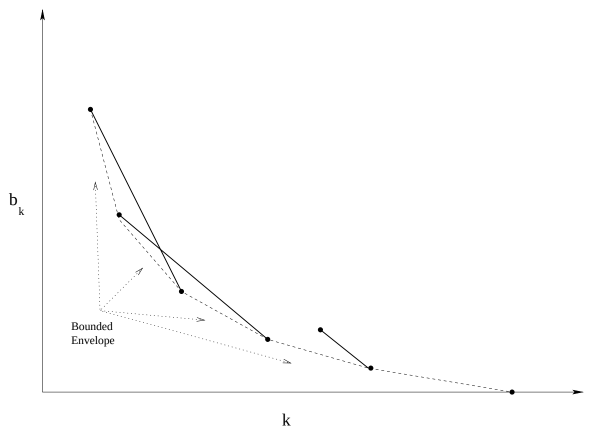

Figure 2.2: Illustration: Bounded Envelope

S_k as points (k, b_k) in a two-dimensional graph with the interpolation of lower bounds, as in Lemma 12, represented as solid lines between (k_1, b_1) and (k_2, b_2) (See Figure 2.2). We also add the points corresponding to the optimal solutions for the values of k for which $OPT_k = 0$. We select a particular set of l solutions for $k = n_1, \ldots, n_l$ such that each point (n_i, b_{n_i}) is on the bounded envelope of the collection of points plotted on the graph. This fixes the values l and n_i for $i = 1, \ldots, l$. We can linearly interpolate the values b_k for $n_{i-1} < k < n_i$ so that property (3) of the properties of a bounded envelope is satisfied.

For values of k for which $OPT_k \geq c_{min}$, the lower envelope is below the solid lines and solid lines are valid lower bound by Lemma 12. For values of k for which $OPT_k = 0$, the bounded envelope coincides with value 0 and hence is a valid lower bound. This guarantees property (1) of the bounded envelope that $b_k \leq OPT_k$ for all k. So we have solutions S_{n_i} for $i = 1, \ldots, l$ and b_k for all k such that all three properties of $(\alpha + \epsilon)$-approximate bounded envelope are satisfied.

Lemma 13 *There exists a 2-approximate bounded envelope for incremental and hierarchical median problems.*

Proof: We can convert the $(2 - \frac{1}{n})$-approximate LMP FL algorithm by Jain et al. [41] to 2-approximate bounded envelope by the above procedure with $\epsilon = \frac{1}{n}$. Finally we can obtain the bounded envelope in the required format by making $\mathsf{ben}(S) = n_f - |S|$. This bounded envelope can be used for both incremental and hierarchical median problems. $\square$

In particular, we obtain a 2-bounded envelope for the k-median problem, and an associated interpolation algorithm for both the incremental and hierarchical k-median problems, which allows us to replace the factors of $(3+\epsilon)$ in the competitive ratios of these problems coming from the approximation algorithm of Arya et al. [6] with a factor of 2.

2.6 Concluding Remarks

Our approach described in Section 2.2, and illustrated in Section 2.3, is general and can be easily used to handle other problems such as the k-center problem and the minimum dominating set problem. In Section 2.3.3, we have considered the incremental facility location problem introduced by [55]. Another natural incremental version of facility location can be defined using a partial facility location problem studied in [20], where all but s cities need to be served. Our approach again obtains an $O(1)$-competitive solution using a $O(1)$-approximation algorithm for the offline version.

One limitation of our work is that it may not lead to the best incremental solutions for a given problem. For instance, we can obtain an efficient 2-competitive algorithm for the unweighted vertex cover problem using the standard primal-dual approach (e.g., [62, Chap. 24]), while our generic approach only achieves a bound of 8. We also mention that for each of the problems discussed in the technical sections, there exists a constant c such that no c-competitive solution exists. For each of these problems, however, the best competitive ratio achievable is not known.

CHAPTER 3

FACILITY LEASING

3.1 Introduction

A current trend in business and logistics is that of a non-asset owning company: for example, a trucking company that owns no trucks, but instead subcontracts the work out to other trucking companies, and makes its profits by efficiently combining the various trucking jobs that it receives [52]. Such organizations rely heavily on optimization to find the efficiencies that make their companies profitable. Some of the optimization problems that arise in such settings have been well-studied in the literature, but some lead to previously unconsidered variations. For instance, in the location theory literature, it is usually assumed that the facilities opened are available to serve customer demand from the moment of opening onwards. However, a non-asset owning company may instead prefer to serve customers by subcontracting or leasing the facilities needed to serve demands at some point in time, then allowing the lease to elapse as demand subsides.

In this chapter, we will consider a formalization of this problem known as the *facility leasing problem*, a variant of the well-studied uncapacitated facility location problem; this variant was introduced by Anthony and Gupta [3]. In this problem, we are given a set F of potential facilities we may lease, and a set D of potential clients. For each potential client $j \in D$ and potential facility $i \in F$, there is a cost c_{ij} associated with serving client j from facility i. We assume these costs obey the triangle inequality, which is a reasonable assumption if the clients and facilities are in a metric space, and the service cost is dependent on the distance between the client and facility. There are distinct time periods, and in each time period t, a set $D_t \subseteq D$ of clients arrives and must be served from some facility open during time period t. There are K different possible lease lengths, $l_1, \ldots, l_K$, and we can lease a facility $i \in F$ starting at any time period t for lease length l_k at a cost f_i^k; that is, the facility i is then open during time units in $[t, t + l_k)$ and can serve any client j arriving during this time interval at cost c_{ij}. Furthermore, the cost of the lease is allowed to depend both on the facility and the lease type. The goal is to minimize the cost of the leases plus the total cost of serving each client in each set D_t from some facility open during time t.

In the *offline* version of the problem, we are given the number of time periods T as input, as well as the set of clients D_t for each time period. In the *online* version of the problem, we do not know the number of time periods, nor do we know the client sets. In each new time period t, we see the client set D_t, and we can sign new leases starting at time t or some time prior (e.g. we can sign a lease for the month of November in mid-November). We then must assign each client in D_t to some facility open at time t.

Since the offline version of the problem contains the uncapacitated facility location problem as a special case, the problem is NP-hard. For the online version of the problem, we use the notion of *competitive ratio*, introduced by Sleator and Tarjan [59], to evaluate our algorithms. We say that an algorithm is α-competitive if its cost after T time steps is no more than α times the value of an optimal solution to the offline problem.

The offline facility leasing problem was introduced by Anthony and Gupta [3] in the context of more general *infrastructure leasing* problems. They show an interesting and surprising connection of infrastructure leasing problems with K

lease types to K-stage stochastic optimization problems. In particular, given an α-approximation algorithm for the K-stage stochastic problem, they derive an $O(\alpha)$-approximation algorithm for the related leasing problem. For the facility leasing problem, therefore, they obtain an $O(K)$-approximation algorithm from an $O(K)$-approximation algorithm for the K-stage stochastic facility location problem due to Swamy and Shmoys [61].[1]

The first result of this chapter is a 3-approximation algorithm for the offline facility leasing problem. To achieve this, we use a modification of the primal-dual algorithm for the uncapacitated facility location problem due to Jain and Vazirani [42]. It is relatively straightforward to give a linear programming relaxation of the facility leasing problem; for the dual of this LP, we increase dual variables uniformly for each client/time pair (j, t) for $j \in D_t$ until a dual constraint associated with a facility lease becomes tight. As with Jain and Vazirani, the trick to obtaining a good approximation algorithm is to open a subset of the tight leases. In our case, however, the aspect of time makes this step of the algorithm slightly more complicated. We resolve this by signing longer leases than the LP indicates; we are then able to obtain the same guarantee as Jain and Vazirani.

We then turn to the online version of the problem. The online facility leasing problem generalizes two problems introduced by Meyerson, *the online facility location problem* [49] and *the parking permit problem* [50]. In *the online facility location problem*, once a facility is opened, it is open for the rest of time; we can consider this as a facility leasing problem with a single lease type, where the lease

[1]Note that [3] erroneously claims that Srinivasan [60] contains an $O(1)$-approximation algorithm for the K-stage stochastic facility location problem, and thus claims an $O(1)$-approximation algorithm for offline facility leasing. A corrected version of the paper is available at `http://www.cs.cmu.edu/~anupamg`.

length is indefinite. Meyerson [49] gives an $O(\log n)$-competitive algorithm for the problem, where n is the number of clients that appear. This is improved by Fotakis [27] to an $O(\frac{\log n}{\log \log n})$-competitive algorithm; furthermore, the author shows that no better competitive ratio is possible for the problem. Another paper of Fotakis [28] gives a simple $O(\log n)$-competitive algorithm for the problem whose choices are guided by solutions to dual of a linear programming relaxation of the problem. Curiously, the competitive ratio of the algorithm is not analyzed in terms of this dual. The algorithm has strong similarities to an algorithm for the (offline) uncapacitated facility location problem of Jain, Mahdian, Markakis, Saberi, and Vazirani [41]. In the *parking permit problem*, a professor walks to work on days when it is sunny, and drives when it rains. If she drives, then she must have a parking permit valid for that day. Parking permits can be purchased for various lengths of time, and given that the weather is unknown, the question is which permits to buy and when. This problem corresponds to the online facility leasing with a single facility i, a single client j, and a service cost $c_{ij} = 0$; buying a parking permit corresponds to leasing the facility i for the corresponding length of time and a rainy day corresponds to a demand of client j in that time period. Meyerson gives a deterministic $O(K)$-competitive algorithm for the parking permit problem, and a randomized $O(\log K)$-competitive algorithm. He also shows that any deterministic algorithm has competitive ratio $\Omega(K)$, and any randomized algorithm has competitive ratio $\Omega(\log K)$.

The second result of this chapter is an $O(K \log n)$-competitive algorithm for the online facility leasing problem. No previous algorithm is known for this problem. We first reanalyze the simple $O(\log n)$-competitive algorithm of Fotakis by using a *dual fitting* argument (first called such by Jain et al. [41], but to our knowledge first used by Chvátal [24]). We show that his algorithm can be viewed as constructing an infeasible solution to the dual of a linear relaxation of the facility leasing problem, such that the cost of the primal solution given by the online algorithm is at most a constant factor times the value of the dual. We then show that scaling the dual variables down by $O(\log n)$ causes it to become feasible, so that the primal cost is at most $O(\log n)$ times the value of a feasible dual, giving the result. We then give a modification of Fotakis' algorithm to the case of facility leasing. Our algorithm is a generalization of both Fotakis' and Meyerson's algorithms in the sense that for the online facility location problem it reduces to Fotakis' algorithm and for the parking permit problem it reduces to Meyerson's algorithm.

This chapter is structured as follows. In Section 3.2, we introduce the linear programming relaxation we will use throughout the chapter, as well as some basic definitions and terminology. In Section 3.3, we give our offline 3-approximation algorithm for the problem. Section 3.4 gives Fotakis' $O(\log n)$-competitive algorithm for the online facility location algorithm and our analysis of it. Section 3.5 gives our extension of Fotakis' algorithm to the online facility leasing problem. We give some open problems in Section 3.6.

3.2 Definitions and Terminology

We have a set of potential facilities locations F which can be opened to serve any subset of clients D which arrive over a period of time from 1 to T. In the offline problem T is part of the input, while for the online problem T is unknown. For every time period $t \in [T]$ every client in the set D_t wants to be served by a facility that is open at time t. A facility can be leased for an interval $[t, t + l_k)$

at a cost of f_i^k for $k \in \mathcal{L}$ where $\mathcal{L}$ is the different set of leases available and any client can be served by the facility leased during this particular interval. For simplicity of notation, we let I_t^k denote the time interval $[t, t + l_k)$. We let $K = |\mathcal{L}|$. We abuse the definition of a facility and say that triple (i, k, t) is a facility at location $i \in F$ leased for a duration of l_k starting at time t. Similarly, a client (j, t) represents $j \in D_t$ seeking a facility from which to be served at time t. For simplicity, we'll denote the set of facility triples (i, k, t) as $\mathcal{F}$, and the set of client demand pairs (j, t) as $\mathcal{D}$. We introduce here some notation we will use throughout the chapter. We use $(a)_+ \equiv \max(0, a)$. For a set $X \subseteq F$ and client $j \in D$, we define $c(X, j) = \min_{i \in X} c_{ij}$. For a set $X \subseteq \mathcal{F}$ and client $(j, t) \in \mathcal{D}$, we define $c(X, (j, t)) = \min_{(i,k,t') \in X, t \in I_{t'}^k} c_{ij}$.

In the offline version of the problem, we seek to find a set of facilities (the facility locations and their leasing intervals) so as to minimize the sum of the cost of opening the facilities and the cost of connecting the clients to the nearest facility open at that time. So we seek to find a set $\mathcal{T} \subseteq \mathcal{F}$ of facilities so as to minimize $\sum_{(i,k,t) \in \mathcal{T}} f_i^k + \sum_{(j,t) \in \mathcal{D}} \min_{(i,k,t') \in \mathcal{T}, t \in I_{t'}^k} c_{ij}$.

In the online version of the problem, we must irrevocably assign each client in D_t to some facility open at time t before we see the clients in D_{t+1}. If we later open another, closer facility to (j, t) than the one it was originally assigned to, we are not allowed to change its assignment.

In both cases, the following linear program is a relaxation of the problem. The variable y_{ikt} indicates whether we open the facility i with lease type k at time t. The variable $x_{ikt',jt}$ indicates whether client (j, t) is assigned to facility

(i,k,t') to be served.

$$\text{Min} \quad \sum_{(i,k,t)\in\mathcal{F}} f_i^k y_{ikt} + \sum_{(j,t)\in\mathcal{D}} \sum_{(i,k,t')\in\mathcal{F}:t\in I_{t'}^k} c_{ij} x_{ikt',jt}$$

subject to:

$$\begin{aligned}
& x_{ikt',jt} \le y_{ikt'} && \forall i,k,t' \\
& \sum_{(i,k,t')\in\mathcal{F}:t\in I_{t'}^k} x_{ikt',jt} \ge 1 && \forall (j,t)\in\mathcal{D} \\
& x_{ikt',jt}, y_{ikt} \ge 0 && \forall (j,t)\in\mathcal{D}, \\
& && (i,k,t),(i,k,t')\in\mathcal{F}.
\end{aligned}$$

Taking the dual, we have

$$\text{Max} \quad \sum_{(j,t)\in\mathcal{D}} v_{jt}$$

subject to:

$$\begin{aligned}
& \sum_{(j,t)\in\mathcal{D}} w_{ikt',jt} \le f_i^k && \forall (i,k,t')\in\mathcal{F} \\
& v_{jt} - w_{ikt',jt} \le c_{ij} && \forall (j,t)\in\mathcal{D}, (i,k,t')\in\mathcal{F}, t\in I_{t'}^k \\
& v_{jt}, w_{ikt',jt} \ge 0 && \forall (j,t)\in\mathcal{D}, (i,k,t')\in\mathcal{F}.
\end{aligned} \tag{3.1}$$

Our algorithms will work by using the dual to construct an integer solution to the primal problem. In the offline case, we construct a feasible dual such that the primal costs no more than 3 times the value of the dual. In the online case, we will construct an infeasible dual solution such that the primal costs no more than $K+1$ times the value of the dual, and scaling the dual variables v_{jt} down by a factor of $O(\log n)$ makes the dual solution feasible. This gives us the two central results of the chapter.

3.3 A 3-Approximation Algorithm for Offline Facility Leasing

In this section, we give a 3-approximation algorithm for the offline facility leasing problem. The approach we follow is similar to the primal-dual facility location algorithm by Jain and Vazirani [42]. The dual LP used is given in Section 3.2.

3.3.1 The Algorithm

As in [42], the algorithm proceeds in two phases. In Phase 1, the algorithm operates in a primal-dual fashion determining a set of temporarily open facilities (triplets) and assigns each client (j, t) to a temporarily open facility. In Phase 2 the algorithm chooses a subset of these facilities to open permanently and reassigns the clients to the permanently open facilities. Phase 1 of our algorithm is simply the extension of [42] to our setting; the main difference of our algorithm is the set of (facility, lease type, time) triplets chosen to open in Phase 2.

Following [42], in the first phase we uniformly increase the dual variables v_{jt} associated with the clients $(j, t) \in \mathcal{D}$. We implicitly maintain the dual variables $w_{ikt',jt} = (v_{jt} - c_{ij})_+$. At some point, we will no longer be able to increase the dual variables and maintain dual feasibility, for one of two reasons. First, some constraint (3.1) will become *tight* for some facility $(i, k, t) \in \mathcal{F}$; that is, the dual constraint is met with equality. In this case we declare the triple (i, k, t) to be temporarily open. Let $\mathcal{T}$ be the set of temporarily open facilities. Second, we might have $v_{jt} = c_{ij}$ for some temporarily open facility (i, k, t') with $t \in I^k_{t'}$; we can't increase v_{jt} further since this would force $w_{ikt',jt} > 0$ and violate the corresponding constraint (3.1). We say that a client (j, t) *contributes to* facility (i, k, t') if $t \in I_{t'}^k$ and $v_{jt} > c_{ij}$; it is *connected to* the facility if $v_{jt} \geq c_{ij}$. After either event happens, we continue increasing the duals of all clients not connected to a temporarily open facility. Eventually, all clients are connected to a temporarily open facility, and Phase 1 ends.

In Phase 1, a client might have contributed towards opening multiple facilities. However we want to ensure that a client contributes only to a single facility lease. Phase 2 ensures this by opening only a subset of these temporarily open facilities. To run Phase 2, we construct a graph $G(V, E)$ with vertex set V as the set of temporarily opened facilities in $\mathcal{T}$ in Phase 1. We add an edge between two facilities in G if there is a client that contributes to both the facilities. As in [42], we now find a maximal independent set in G; here, however, we give priority for facilities with longer lease length to be in the independent set. In other words, we order the temporarily open facilities according to non-increasing lease lengths then greedily add facilities to the independent set following this order. This gives an independent set $\mathcal{I} \subseteq \mathcal{T}$ with the following properties:

1. The independent set is maximal.
2. For every temporarily opened facility not in the independent set there is a facility in the independent set with same or longer lease length adjacent to it in the graph G.

Given the set $\mathcal{I}$, for each triple $(i, k, t) \in \mathcal{I}$, we sign three leases, the one corresponding to (i, k, t), then the two leases at facility i of type k that start at time $t + l_k$ and end at time t; that is, we open (i, k, t), $(i, k, t + l_k)$, and $(i, k, (t - l_k)_+)$. Let the set of facilities opened be $\mathcal{I}'$. Note that $|\mathcal{I}'| = 3 \cdot |\mathcal{I}|$.

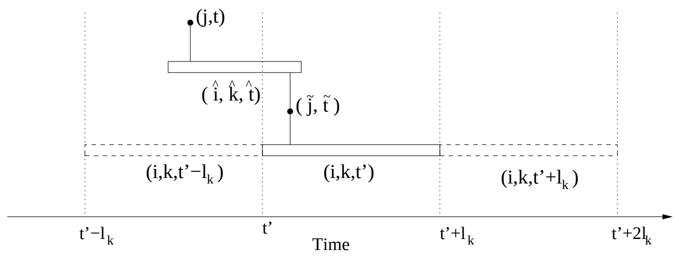

Figure 3.1: Illustration: A client indirectly connected to a leased facility

3.3.2 The Analysis

Consider any client (j, t). If it is connected to a facility $(i, k, t') \in \mathcal{I}$ then we assign (j, t) to that facility and say that (j, t) is *directly* connected to (i, k, t'). If it is not connected to a facility in $\mathcal{I}$, then since (j, t) connected to some temporarily open facility $(\hat{i}, \hat{k}, \hat{t})$, and $\mathcal{I}$ is a maximal independent set in G, this temporarily open facility must be adjacent to some $(i, k, t') \in \mathcal{I}$ via an edge of G. We will *indirectly* connect client (j, t) to one of the three facilities opened corresponding to (i, k, t'). We claim that $t \in [(t' - l_k)_+, t' + l_k)$, so that one of these facilities can serve (j, t). See Figure 3.1 for an illustration. To prove the claim, observe that since there is an edge between $(\hat{i}, \hat{k}, \hat{t})$ and (i, k, t') in G, there is some client $(\tilde{j}, \tilde{t})$ that contributes to both facilities. This implies that $\tilde{t} \in I_{\hat{t}}^{\hat{k}} \cap I_{t'}^{k}$. Furthermore, by property (2) of $\mathcal{I}$, the length of the lease $l_{\hat{k}}$ of the temporarily opened facility $(\hat{i}, \hat{k}, \hat{t})$ is no longer than the lease l_k. Thus the interval $I_{\hat{t}}^{\hat{k}} \subseteq [(t' - l_k)_+, t' + l_k)$. Since (j, t) connects to $(\hat{i}, \hat{k}, \hat{t})$, then $t \in I_{\hat{t}}^{\hat{k}} \subseteq [(t' - l_k)_+, t' + l_k)$, and our claim is proven.

We show that three times the sum of dual variables of the clients pays for the facility leasing costs and the cost of serving clients from the nearest open facility. For each client (j, t) directly connected to a facility (i, k, t'), let $v_{jt}^f = v_{jt} - c_{ij}$ and let $v_{jt}^s = c_{ij}$. For each client (j,t) indirectly connected to a facility (i,k,t'), let $v_{jt}^f = 0$ and $v_{jt}^s = v_{jt}$. Note that $v_{jt} = v_{jt}^f + v_{jt}^s$ for all clients (j,t).

Lemma 14 *For each facility (i,k,t') in $\mathcal{I}$,*

$$\sum_{(j,t) \textit{ directly connected to } (i,k,t')} v_{jt}^f = f_i^k.$$

Proof: Note that this facility (i,k,t') has a tight dual inequality in Phase I. Since this facility belongs to the independent set $\mathcal{I}$, all the clients that contributed to opening this facility are directly connected to this facility. So the sum of all these contributions of directly connected clients is equal to the facility opening cost. $\square$

Corollary 1

$$\sum_{(i,k,t)\in\mathcal{I}} f_i^k = \sum_{(j,t)\in\mathcal{D}} v_{jt}^f.$$

Lemma 15

$$\sum_{(i,k,t)\in\mathcal{I}'} f_i^k = 3 \cdot \sum_{(j,t)\in\mathcal{D}} v_{jt}^f.$$

Proof: The opening cost of the facilities in $\mathcal{I}'$ is at most three times the opening cost of facilities in $\mathcal{I}$. By Corollary 1 this is at most $3 \cdot \sum_{(j,t)\in\mathcal{D}} v_{jt}^f$. $\square$

Lemma 16 *For every client (j,t) indirectly connected to a facility (i,k,t'), the connection cost $c_{ij} \leq 3 \cdot v_{jt}^s$.*

Proof: Since (j,t) is indirectly connected to (i,k,t'), there should be an edge in G between (i,k,t') and some temporarily opened $(\hat{i},\hat{k},\hat{t})$ to which (j,t) is connected. Because there is an edge between (i,k,t') and $(\hat{i},\hat{k},\hat{t})$, there is a client

$(\tilde{j}, \tilde{t})$ with positive contributions to the facility opening costs of both these facilities. Since $(\tilde{j}, \tilde{t})$ contributes to both the facilities, $c_{\tilde{i}\tilde{j}} \leq v_{\tilde{j}\tilde{t}}$ and $c_{\hat{i}\tilde{j}} \leq v_{\tilde{j}\tilde{t}}$. Note that $v_{\tilde{j}\tilde{t}}$ will stop increasing when one of the two facilities is temporarily open and that since (j, t) connected to $(\hat{i}, \hat{k}, \hat{t})$, v_{jt} stopped increasing at a point no earlier than when $(\hat{i}, \hat{k}, \hat{t})$ was temporarily opened. Thus $v_{jt} \geq v_{\tilde{j}\tilde{t}}$ and by triangle inequality $c_{ij} \leq c_{\hat{i}j} + c_{\hat{i}\tilde{j}} + c_{i\tilde{j}} \leq v_{jt} + v_{\tilde{j}\tilde{t}} + v_{\tilde{j}\tilde{t}} \leq 3 \cdot v_{jt} = 3 \cdot v^s_{jt}$. □

Let $y_{ikt} = 1$ for each $(i, k, t) \in \mathcal{I}'$ and $y_{ikt} = 0$ otherwise. For each client (j, t) let (i, k, t') be the facility in $\mathcal{I}'$ which it is connected to, directly or indirectly. Set $x_{ikt',jt} = 1$ for all such client-facility pair. Note that $(\mathbf{x}, \mathbf{y})$ is primal feasible. The theorem below follows directly from the preceding discussion.

Theorem 19 *The primal feasible solution (x, y) and the dual feasible solution (v, w) satisfy*

$$\sum_{(i,k,t)\in\mathcal{F}} f^k_i y_{ikt} + \sum_{(j,t)\in\mathcal{D}} \sum_{(i,k,t)\in\mathcal{F}: t\in I^k_{t'}} c_{ij} x_{ikt',jt} \leq 3 \cdot \sum_{(j,t)\in\mathcal{D}} v_{jt} \leq 3 \cdot OPT.$$

Thus the algorithm is a 3-approximation algorithm for the offline facility leasing problem.

Proof: For each client (j, t) directly connected to a facility (i, k, t'), its service cost $c_{ij} = v^s_j$. By Lemma 16, if (j, t) is indirectly connected to a facility (i, k, t'), its service cost $c_{ij} \leq 3v^s_j$. By Lemma 15, the total cost of leases signed is $3\sum_{(j,t)\in\mathcal{D}} v^f_{jt}$. Therefore, the total service cost plus the total leasing cost is at most

$$3 \sum_{(j,t)\in\mathcal{D}} \left(v^s_{jt} + v^f_{jt}\right) = 3 \sum_{(j,t)\in\mathcal{D}} v_{jt} \leq 3 \cdot \mathrm{OPT},$$

where the final inequality follows by weak duality and the feasibility of (v, w) for the dual linear program in Section 3.2. □

3.4 Fotakis' Online Facility Location Algorithm

In this section, we give Fotakis' algorithm for the online facility location problem, and restate the analysis of it as a dual-fitting argument. The algorithm will construct an infeasible dual solution to the dual of Section 3.2 such that the cost of the primal solution constructed is at most twice the dual objective. We'll then show that scaling the dual variables down by a factor of $O(\log n)$ will give a feasible dual solution, yielding the competitive ratio of the algorithm. In the next section, we'll show how a modification of Fotakis' algorithm and this analysis gives our result for the online facility leasing problem.

3.4.1 The Algorithm

Note that in the case of the online facility location problem, we have a single facility lease type and its duration is infinite; once a facility is opened it continues to remain open. We denote the cost of the facility at i by f_i.

The algorithm works as follows. We maintain an (infeasible) set of dual variables v_{jt}, one for each client $j \in D_t$ for all time periods thus far. We also maintain a set X of facilities that have been opened so far, which is initially empty. Each client (j, t) that has arrived at some prior point in time and been connected to some facility in X bids $(c(X, j) - c_{ij})_+$ towards the facility cost of each unopened facility i. When a set of clients D_t arrives, we sequence through the clients $j \in D_t$ in index order. We increase the dual variable v_{jt}; client (j, t) bids $(v_{jt} - c_{ij})_+$ towards the cost of each unopened facility i. We continue increasing v_{jt} until either $v_{jt} = c_{i'j}$ for some previously opened facility $i' \in X$, or the sum of the bids on some unopened facility i is equal to its facility cost f_i. In the former case, we assign client (j,t) to the facility i'; in the latter case, we open facility i, add i to X, and assign (j,t) to i. We then continue to the next facility in D_t. Note that client (j,t) immediately reduces its bids on other unopened facilities $\hat{i}$ from $(v_{jt} - c_{\hat{i}j})_+$ to $(c(X,j) - c_{\hat{i}j})_+ = (c_{ij} - c_{\hat{i}j})_+$. The algorithm is summarized in Algorithm 4. The algorithm is very similar to an algorithm of Jain et al. [41] for the (offline) uncapacitated facility location problem, except that in that algorithm all clients increase their dual variables simultaneously (since all are known in advance), and clients whose bids are used to open an unopened facility are then reassigned to that facility.

Algorithm 4: Fotakis' algorithm for online facility location

```
$X \leftarrow \emptyset; D \leftarrow \emptyset; t \leftarrow 0$
While true
    For each $j \in D_t$
        Increase $v_{jt}$ until $v_{jt} = c_{ij}$ for $i \in X$ or $(v_{jt} - c_{ij})_+$
            $+ \sum_{(j',t') \in D} (c(X,j') - c_{ij'})_+ = f_i$ for $i \notin X$
        Assign $(j,t)$ to $i$; $X \leftarrow X \cup \{i\}$; $D \leftarrow D \cup \{(j,t)\}$
    $t \leftarrow t + 1$
```

3.4.2 The Analysis

We first analyze the cost of the primal solution.

Lemma 17 *The cost of the solution produced by the algorithm is at most* $2\sum_{(j,t)\in\mathcal{D}} v_{jt}$.

Proof: We show that both the service costs and the facility costs of the solution are each bounded above by $\sum_{(j,t)\in\mathcal{D}} v_{jt}$, which will give the lemma. Note that when a client (j,t) is assigned to a facility i, either it had made a bid on that facility and caused it to open, in which case $(v_{jt} - c_{ij})_+ > 0$, implying $v_{jt} > c_{ij}$, or i was already open and $v_{jt} = c_{ij}$. In either case, v_{jt} is at least the service cost c_{ij}.

To bound the facility costs $\sum_{i\in X} f_i$, we note that when we opened a facility i, its cost was equal to the sum of the bids on that facility. We show that for a given client (j,t), the sum of its accepted bids is at most v_{jt}, which implies that $\sum_{i\in X} f_i \leq \sum_{(j,t)\in\mathcal{D}} v_{jt}$. In particular, we show that once the bid of client (j,t) is used to open some facility i, all its other outstanding bids are reduced by the amount bid towards i; since it bids at most v_{jt} towards any facility at any point in time, this is sufficient to prove the claim. Consider two facilities, i and i'. Before (j,t) is connected to any facility, it bids $(v_{jt} - c_{ij})_+$ towards the first and $(v_{jt} - c_{i'j})_+$ towards the latter. If facility i is opened and (j,t) is assigned to i, then (j,t) reduces its bid for i' to $(c(X,j) - c_{i'j})_+ = (c_{ij} - c_{i'j})_+$; that is, it reduced its bid for i' by $v_{jt} - c_{ij}$, exactly the bid accepted for opening facility i. Similarly, if (j,t) is assigned to some facility, it bids towards unopened facilities i and i' $(c(X,j) - c_{ij})_+$ and $(c(X,j) - c_{i'j})_+$ respectively. If its bid towards i is accepted, and i is opened, then it must have been the case that $(c(X,j) - c_{ij})_+ > 0$, so that i is closer to j than any other facility in X. Once i is opened, it is added to X so that the bid to i' becomes $(c(X \cup i, j) - c_{i'j})_+ = (c_{ij} - c_{i'j})_+$. The bid towards i' is reduced by $c(X,j) - c_{ij}$, exactly the bid accepted towards i. $\square$

The following lemma will be useful in bounding bids towards facilities in the remainder of the proof.

Lemma 18 *Consider clients (j,t) and (l,t'), such that j is considered before l, so that we increase the dual of (j,t) before that of (l,t'). Then for X open when we increase $v_{lt'}$, for any facility i, $c(X,j) - c_{ij} \geq v_{lt'} - c_{il} - 2c_{ij}$.*

Proof: Consider the facility $h \in X$ to which j is closest at the point we increase the dual for (l,t'). The dual value $v_{lt'}$ is no more than c_{hl}, since at this point h is open and we could have assigned (l,t') to h. By triangle inequality $v_{lt'} \leq c_{hl} \leq c_{il} + c_{ij} + c_{hj}$. So $c(X,j) - c_{ij} = c_{hj} - c_{ij} \geq v_{lt'} - c_{il} - 2c_{ij}$. $\square$

Let H_n be the nth harmonic number $1 + \frac{1}{2} + \cdots + \frac{1}{n}$. Our goal is now to show that for $\alpha = 1/2H_n$, and for any facility $i \in F$, $\sum_{(j,t)\in\mathcal{D}}(\alpha v_{jt} - c_{ij})_+ \leq f_i$. Thus scaling down the dual solution v by $2H_n$ gives a feasible solution to the dual program in Section 3.2. To prove this, we show the following lemma.

Lemma 19 *For any $S \subseteq \mathcal{D}$ and any facility i, $\sum_{(j,t)\in S}(\alpha v_{jt} - c_{ij}) \leq f_i$.*

Proof: For ease of exposition, we let $S = \{1, 2, \ldots, p\}$, dropping the pair notation for clients, with the understanding that we increase the dual variables for the clients in the order of the indices. Consider any $l \in S$. At the point in time at which we increase the dual for l, the total bids for facility i are $(v_l - c_{il})_+ + \sum_{j<l}(c(X,l) - c_{ij})_+ \leq f_i$.

So we have

$$\begin{aligned}
f_i &\geq (v_l - c_{il})_+ + \sum_{j<l}(c(X,j) - c_{ij})_+ \\
&\geq (v_l - c_{il}) + \sum_{j<l}(c(X,j) - c_{ij}) \\
&\geq (v_l - c_{il}) + \sum_{j<l}(v_l - c_{il} - 2c_{ij}) \\
&\geq l(v_l - c_{il}) - 2\sum_{j<l} c_{ij},
\end{aligned}$$

where the penultimate inequality follows from Lemma 18. Dividing both sides by l we get

$$\frac{1}{l} f_i \geq (v_l - c_{il}) - \frac{2}{l} \sum_{j<l} c_{ij}.$$

Observing that

$$\sum_{l=1}^{p} \frac{2}{l} \sum_{j<l} c_{ij} = \sum_{l=1}^{p} 2c_{il}(H_p - H_l), \tag{3.2}$$

and adding the previous inequality for all l in S gives

$$\begin{aligned} H_p f_i &\geq \sum_{l=1}^{p} (v_l - c_{il}) - \sum_{l=1}^{p} 2c_{il}(H_p - H_l) \\ &= \sum_{l=1}^{p} (v_l - 2c_{il}H_p) + \sum_{l=1}^{p} 2c_{il} \left(H_l - \frac{1}{2} \right) \\ &\geq \sum_{l=1}^{p} (v_l - 2c_{il}H_p) \end{aligned}$$

Dividing by $2H_p$ we get $\sum_{l=1}^{p} \left(\frac{v_l}{2H_p} - c_{il} \right) \leq \frac{f_i}{2} \leq f_i$. Since $H_p \leq H_n$, the lemma statement holds. □

Corollary 2 *αv is a dual feasible solution.*

Proof: Applying the lemma above with $S = \{(j,t) \in \mathcal{D} : v_{jt} - c_{ij} > 0\}$ proves that $\sum_{(j,t)\in\mathcal{D}} (\alpha v_{jt} - c_{ij})_+ \leq f_i$. If we set $w_{ikt',jt} = (\alpha v_{jt} - c_{ij})_+$, then $(\alpha \mathbf{v}, \mathbf{w})$ is a feasible solution to the dual. □

The theorem below follows from the previous discussion.

Theorem 20 *Fotakis' algorithm gives a $4H_n$-competitive algorithm for the online facility location problem.*

Proof: By Lemma 17, the created primal solution costs at most $2\sum_{(j,t)\in\mathcal{D}} v_{jt}$. By Corollary 2, $v/2H_n$ is dual feasible, so that $\sum_{(j,t)\in\mathcal{D}} v_{jt} \leq 2H_n \cdot \text{OPT}$. The theorem statement follows. □

3.5 An $O(K \log n)$-Competitive Online Facility Leasing Algorithm

In this section we modify Fotakis' algorithm to give a $O(K \log n)$-competitive algorithm for online facility leasing problem. Our algorithm constructs a dual infeasible solution and a primal feasible integral solution that costs no more than $(K + 1)$ times the dual objective function. Then we show that by scaling down the duals by a factor of $O(\log n)$ we get a dual feasible solution yielding the required competitive ratio. The main difference of our algorithm is that clients use their dual variables to bid on the K different lease types at the same time, and bids are reduced on leases of length k only as a lease of type k is opened. At a very high level, the factor of $O(K)$ comes from clients contributing to each lease type simultaneously, and the $O(\log n)$ from the underlying online facility location problem.

3.5.1 The Algorithm

First, following Meyerson [50] and Anthony and Gupta [3], it will be useful to change the leases available to be somewhat more structured.

Lemma 20 (Lemma 2 [3]) *Given an instance I of the facility leasing problem, it can* *be converted into another instance I' of the facility leasing problem such that leases of type k are only available at times t divisible by l_k such that any solution to I can be converted into a solution to I' that costs no more than twice as much.*

In particular, this implies an $O(\alpha)$-competitive algorithm for the instances with leases of this structure is $O(\alpha)$-competitive for the general problem. From here on, we assume that leases are structured in this way.

Following the ideas of the previous section, we maintain a set of dual variables v_{jt} for each $j \in D_t$. We also maintain a set of facilities X opened so far and set of facilities X^k of lease length k opened so far. Every client (j, t) that has arrived at some prior time bids $(\min[v_{jt}, c(X^k, j)] - c_{ij})$ towards a facility (i, k, t') if $t \in I^k_{t'}$. We maintain the invariant that the sum of the bids of all the clients seen so far to a facility is no more than the facility's opening cost.

When a new client arrives, we increase the dual of that client until either the total bids towards some unopened facility is equal to its facility cost, or the client's dual is equal to its cost to receive service from an already open facility, whichever occurs earlier. In first case, we open the facility, connect the client to it, and reduce the bids of all clients contributing to this facility to other unopened facilities of the same lease type k by the bid value of the client. In the second case, the client is connected to the open facility. We then repeat these steps for each of the arriving clients.

Algorithm 5: Online Leasing Algorithm

```
$X \leftarrow \emptyset; X^k \leftarrow \emptyset; D \leftarrow \emptyset; t \leftarrow 0$
While true
    For each $j \in D_t$
        Increase $v_{jt}$ until $v_{jt} = c_{ij}$ for $(i,k,\hat{t}) \in X$ and $t \in I^k_{\hat{t}}$ or $(v_{jt} - c_{ij})_+$
            $+ \sum_{(j',t') \in D, t' \in I^k_{\hat{t}}} (\min[v_{j't'}, c(X^k, j')] - c_{ij'})_+ = f_i$
            for $(i,k,\hat{t}) \notin X$ and $t \in I^k_{\hat{t}}$
        Assign $(j,t)$ to $(i,k,\hat{t})$;
        $X \leftarrow X \cup \{(i,k,\hat{t})\}$;
        $X^k \leftarrow X^k \cup \{(i,k,\hat{t})\}$;
        $D \leftarrow D \cup \{(j,t)\}$
    $t \leftarrow t + 1$
```

3.5.2 The Analysis

Here we show that for $\alpha = \frac{1}{2(H_n+1)}$, $\sum_{(j,t) \in D, t \in I^k_{t'}} (\alpha v_{jt} - c_{ij})_+ \leq f^k_i$ for any facility (i,k,t').

Lemma 21 *For any $S \subseteq \mathcal{D}$ and any facility (i,k,t'), $\sum_{(j,t) \in S, t \in I^k_{t'}} (\alpha v_{jt} - c_{ij}) \leq f^k_i$.*

Proof: Without loss of generality we assume that S consists only of clients (j,t) such that $t \in I^k_{t'}$ as any other client $(j,t) \in S$ such that $t \notin I^k_{t'}$ will not contribute to the summation. We say a client (j,t) is connected to an open facility $(\hat{i},k,\hat{t})$ if $v_{jt} \geq c_{\hat{i}j}$ and $t \in I^k_{\hat{t}}$. Let $S = \{1,2,\ldots,p\}$ where we index the clients in S according to the order in which they became connected to a open facility of lease length k in the algorithm. If there are clients which are connected to a facility of lease type k in the same iteration of the algorithm, then we order them according to non-increasing order of their $v_{jt} - c_{ij}$. Note that we have dropped the pair notation for the clients for ease of exposition.

Consider any client $l \in S$ which was connected to a open facility of lease level k at some point in time. Let X^k be the set of facilities of k^{th} lease level that are open at the start of the iteration in which l is first connected to an open facility of k^{th} level lease. Let $h \leq l$ be the first client in the ordering that was connected to a k^{th} level lease in the same iteration as the client l. Consider the invariant for the facility (i, k, t') at the time the client l became connected to a k^{th} lease type.

$$\begin{aligned}
f_i^k &\geq \sum_{h \leq j \leq l} (v_j - c_{ij})_+ + \sum_{j<h} (c(X^k, j) - c_{ij})_+ \\
&\geq \sum_{h \leq j \leq l} (v_j - c_{ij}) + \sum_{j<h} (c(X^k, j) - c_{ij}) \\
&\geq (l - h + 1)(v_l - c_{il}) + \sum_{j<h} (v_l - c_{il} - 2c_{ij}) \qquad (3.3) \\
&\geq l(v_l - c_{il}) - 2\sum_{j<l} c_{ij} \qquad (3.4)
\end{aligned}$$

Inequality (3.3) follows from the following claim and our ordering of the set S; we know for all j with $h \leq j \leq l$, $v_j - c_{ij} \geq v_l - c_{il}$.

Claim 1 *For any client $j \in S$ which is connected to a k^{th} level facility in an earlier iteration than when $l \in S$ is connected to a k^{th} level facility, $c(X^k, j) - c_{ij} \geq v_l - c_{il} - 2c_{ij}$.*

Dividing Inequality (3.4) by l we get $\frac{1}{l} f_i^k \geq (v_l - c_{il}) - \frac{2}{l} \sum_{j<l} c_{ij}$. Let q be the least index in S such that for all $j > q$, client j does not connect to any k^{th} level lease while the lease for (i, k, t') is available. Then we know that the invariant at the end of the lease gives $\sum_{j \in S: j > q}(v_j - c_{ij}) \leq f_i^k$. Adding this inequality and the prior inequality for all $l \in \{1, \ldots, q\}$, we get

$$\begin{aligned}
(H_q + 1)f_i &\geq \sum_{l=1}^{p}(v_l - c_{il}) - \sum_{l=1}^{q} \frac{2}{l} \sum_{j<l} c_{ij} \\
&= \sum_{l=1}^{p}(v_l - c_{il}) - \sum_{l=1}^{q} 2c_{il}(H_q - H_l) \\
&\geq \sum_{l=1}^{p}(v_l - 2c_{il}H_q) + \sum_{l=1}^{q} 2c_{il}\left(H_l - \frac{1}{2}\right) \\
&\geq \sum_{l=1}^{p}(v_l - 2c_{il}H_q) \\
&\geq \sum_{l=1}^{p}(v_l - 2c_{il}(H_q + 1)),
\end{aligned}$$

where the first equation uses (3.2). Dividing by $2(H_q + 1)$ we get

$$\sum_{l=1}^{p}\left(\frac{v_l}{2(H_q + 1)} - c_{il}\right) \leq \frac{f_i^k}{2} \leq f_i^k$$

which proves dual feasibility for $\frac{v_l}{2(H_n+1)}$ since $H_q \leq H_n$.

Proof of Claim 1: Consider the facility $h \in X^k$ that j is connected to such that $c(X^k, j) = c_{hj}$. Note that the lease duration of this facility is $I_{t'}^k$ because of our assumption on the lease structure of the facilities in Lemma 20 and the assumption that all the clients in S arrived during $I_{t'}^k$. We claim that the dual value v_l is no more than c_{hl}. If l arrived at a time after the iteration in which j became connected to h, then on l's arrival, h was open and so it must be the case that $v_{lt} \leq c_{hl}$. Now suppose l arrived before j is connected to h, or at the same iteration. If $v_l - c_{hl} > 0$, then since l is contributing to the k^{th} level facility h, l would have connected to h at the same iteration as j, contradicting the hypothesis of the lemma that l was not connected to a k^{th} level facility when h was opened. Thus $v_l \leq c_{hl}$. By triangle inequality $v_l \leq c_{hl} \leq c_{il} + c_{ij} + c_{hj}$. So $c(X^k, j) - c_{ij} = c_{hj} - c_{ij} \geq v_l - c_{il} - 2c_{ij}$. □

Lemma 22 *The cost of opening facilities in X is no more than K times the sum of the duals of the clients.*

Proof: The proof here follows that of Lemma 17, and the observation that whenever a client's bid for a facility of lease type k is accepted, its bid for all other facilities of lease type k is reduced by the amount of its accepted bid. $\square$

Lemma 23 *The sum of the connection costs of the clients to its assigned facilities is no more than the sum of the duals.*

Proof: As in the proof of Lemma 17, when a client is assigned to a facility by the algorithm, its dual is no less than the service cost for being assigned to that facility. $\square$

Corollary 3 *αv is a dual feasible solution for $\alpha = 1/(2H_n + 1)$.*

Proof: For any facility $(i, k, t') \in \mathcal{F}$, applying Lemma 21 to the set $S = \{(j,t) \in \mathcal{D} : v_{jt} - c_{ij} > 0, t \in I^k_{t'}\}$ proves that $\sum_{(j,t)\in\mathcal{D},t\in I^k_{t'}}(\alpha v_{jt} - c_{ij})_+ \leq f^k_i$. If we set $w_{ikt',jt} = (\alpha v_{jt} - c_{ij})_+$, then $(\alpha\mathbf{v}, \mathbf{w})$ is a feasible solution to the dual. $\square$

Theorem 21 *The online facility leasing algorithm is a $O(K \log n)$-competitive algorithm.*

Proof: By Lemmas 22 and 23, the cost of the primal solution constructed by the algorithm is at most $(K+1)\sum_{(j,t)\in\mathcal{D}} v_{jt}$. By Corollary 3 $\frac{v}{2(H_n+1)}$ is dual feasible, so that $\sum_{(j,t)\in\mathcal{D}} v_{jt} \leq 2(H_n+1)\cdot\text{OPT}$. This implies that the algorithm is $2(H_n+1)(K+1)$-competitive for instances in which the leases are structured as in Lemma 20. Finally, by Lemma 20, any $O(\alpha)$-competitive algorithm for leases structured in the way specified by the lemma is $O(\alpha)$-competitive for the original instance. $\square$

3.6 Conclusion

The most interesting open question arising from this work is whether the factor $O(K \log n)$ is nearly tight for a deterministic algorithm. It is possible that the $\Omega(K)$ deterministic bound of Meyerson [50] for the parking permit problem and the $\Omega(\frac{\log n}{\log \log n})$ bound of Fotakis [27] for the online facility location problem can be combined to give a deterministic $\Omega(K \frac{\log n}{\log \log n})$ lower bound on the competitive ratio of the facility leasing problem. Potentially, however, much better bounds are possible. In particular, it would be interesting to consider randomized online algorithms for the problem.

Given that we can achieve a 3-approximation algorithm for the offline problem by a simple modification of an uncapacitated facility location algorithm, it would be interesting to see if other, better approximation algorithms for the problem can similarly be adapted in order to provide improved performance guarantees.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Introduction

In this chapter, we consider some of the existing k-median, incremental k-median and hierarchical k-median algorithms and compare the running times and the quality of solutions against our incremental and hierarchical k-median algorithms using some of the existing benchmark datasets for the k-median problem.

4.2 Algorithms

4.2.1 The k-median problem

In this section we consider five different algorithms for the k-median problem. The first one is the single swap local search algorithm by Arya et al. [6], which gives a solution which costs within 5 times the cost of the optimal solution. We also consider Charikar et al.'s [19] linear program (LP) rounding algorithm which rounds the LP optimum to get an integer solution which is no more than 8 times the cost of the optimal LP solution. Jain et al. [41] give a greedy dual-fitting Lagrangean Multiplier Preserving (LMP) Facility Location (FL) which we use to obtain a bounded envelope for the k-median problem as described in Section 2.5.2. We also consider the standard k-median linear program and solve it optimally using CPLEX. The optimal solution can be fractional but still gives a good lower bound for the k-median problem. We also solve the k-median integer program optimally using CPLEX even though the algorithm is not polynomial time.

Local Search Algorithm of Arya et al.

We consider the Arya et al.'s ([6]) single swap local search algorithm which guarantees a solution within a factor of 5 from the optimal solution. The local search algorithm proceeds by starting with an arbitrary solution and repeatedly doing *valid swaps* on the current solution till no more valid swaps exist. A swap closes a facility in the current solution and opens a facility that was previously closed. A swap is considered valid if the cost of the new solution after swapping is lesser than the cost of the solution before swapping.

Arya et al. proved that the local search algorithm can be made polynomial to run in time polynomial in the input size by considering swaps as valid only if it improves the cost of the solution by a value that is polynomial in the input size. However, for simplicity, we consider any cost improving swap as a valid swap. We run this local search algorithm for each cardinality k. After this procedure we have locally optimal solutions for each value of k.

We do not implement the multi-swap (swaps involving more than one facilities) local search algorithm by Arya et al. because of its high running time even though it gives better approximation guarantee of $3 + \epsilon$. We use the locally optimal solution of cardinality $k - 1$ as a starting solution for the local search iteration for cardinality k. Since this solution is already a good solution for cardinality k we reduce the running times of the subsequent iterations. On average

Algorithm 6: Local Search

```
1. $S \leftarrow$ an arbitrary feasible solution
2. While $\exists S'$ such that $S'$ can be obtained by a single facility swap of $S$ and $cost(S') < cost(S)$
       do $S \leftarrow S'$
3. return S
```

this improves the running times of local search by about 40%.

LP rounding algorithm of Charikar et al.

We consider the LP rounding algorithm of Charikar et al. [19] which takes in the fractional optimal solution of the standard LP relaxation $(k-P)$ of the k-median problem and produces an integer solution that is no more than 8 times the cost of LP optimum.

$$
(k-P) \qquad
\begin{array}{lll}
\text{Min} & \displaystyle\sum_{i\in F, j\in C} c_{ij}x_{ij} & \\
\text{subject to:} & & \\
& \displaystyle\sum_{i\in F} x_{ij} = 1 & \forall j \in C \\
& x_{ij} \le y_i & \forall i \in F, \forall j \in C \\
& \displaystyle\sum_{i\in F} y_i \le k & \\
& x_{ij} \ge 0 & \forall i \in F, \forall j \in C \\
& y_i \ge 0 & \forall i \in F.
\end{array}
$$

The algorithm consists of three steps. We start with the optimal LP fractional solution for a particular value of k.

1. We simplify the problem instance by consolidating nearby locations and combining their demands such that the locations with nonzero demands are far from each other the resulting problem instance.
2. We then simplify the structure of the optimal fraction solution by consolidating nearby fractional centers. The resulting solution has nonzero fractional y value only on facilities with nonzero demands and their y values are no less than $\frac{1}{2}$. We then modify this solution to a $\{0, \frac{1}{2}, 1\}$ solution where the y values take values of only 0, $\frac{1}{2}$ and 1.
3. We then open no more than k of these facilities with non-zero y values selecting based on closeness to other facilities with positive y values.

This algorithm is explained in more detail in Algorithm 7.

Greedy LMP FL Algorithm of Jain et al.

Jain et al. [41] give a LMP greedy dual-fitting FL algorithm for the facility location problem. In this algorithm, we maintain a dual value v_j for every client which is its total contribution to getting connected to a open facility. Some part of this dual v_j pays for the j's connection cost and the remainder is paid toward facility opening costs. We increase the duals of the clients uniformly and open a facility when a facility has enough contribution from the clients to match the facility opening cost. We say a client is connected to a facility if the connection cost is paid for by its dual value. We stop increasing the dual for a client if

Algorithm 7: LP rounding of Charikar et al. [19]

1. Let $(\overline{x}, \overline{y})$ be the optimal LP solution to $(k-P)$. Let the demands be 1 for all locations.

2. Step 1: Let the clients be indexed in the increasing order of their objective function contribution i.e. $\overline{C}_1 \leq \overline{C}_2 \leq \ldots \leq \overline{C}_n$ where $\overline{C}_j = \sum_{i \in N} c_{ij}\overline{x}_{ij}$.

```
    For $j = 1$ to $n$
        If there is another location $i < j$ such that the demand $d_i > 0$ and
        $c_{ij} \leq 4\overline{C}_j$ and set $d_i \leftarrow d_i + d_j$ and $d_j \leftarrow 0$.
    Let $N = \{r : d_r > 0\}$.
```

3. Step 2: Set $y'_j = \overline{y}_j$ for all locations j.
   For each location i such that $y'_i > 0$ and $i \notin N$ let $j \in N$ be its closest location in N. Set $y'_j \leftarrow \min(1, y'_i + y'_j)$ and $y'_j \leftarrow 0$.
   Let $s(j)$ be the closest location to j in N (other than j and ties broken with smallest index). Let $n' = |N|$.
   Sort the locations in N in the decreasing order of $d_j c_{s(j)j}$. Set $\hat{y} = 1$ for the first $2k - n'$ locations and $\hat{y} = \frac{1}{2}$ for the remaining $2(n' - k)$ locations.

4. We build a collection of trees as follows: For each node $i \in N$ with $\hat{y}_i = \frac{1}{2}$ draw a directed edge from i to $s(i)$. Delete one arbitrary edge in every directed (2-)cycle in this graph. This graph is now a collection of rooted trees. Define the level of any node to be the number of edges on the path from the node to the root.
   We select the nodes with $\hat{y} = 1$ in our k-median solution. We partition the nodes $\{i \in N | \hat{y} = \frac{1}{2}\}$ into two subsets corresponding to odd and even levels and include the smaller of the two subsets in our k-median solution. This ensures that there are no more than k facilities in the solution.

it is connected to a open facility. The algorithm is explained in more detail in Algorithm 8.

Algorithm 8: Jain et al.'s Greedy Facility Location Algorithm

1. There is a notion of time. The algorithm starts at time 0. At this time, each client is defined to be unconnected (U:=C), all facilities are unopened, and v_j is set to 0 for every client j. At every moment, each client j offers some money from its contribution to each unopened facility i. The amount of this offer is computed as follows: If j is unconnected, the offer is equal to $max(v_j - c_{ij}, 0)$ (i.e., if the contribution of j is more than the cost that it has to pay to get connected to i, it offers to pay this extra amount to i); If j is already connected to some other facility i', then its offer to facility i is equal to $max(c_{i'j} - c_{ij}, 0)$ (i.e., the amount that j offers to pay to i is equal to the amount j would save by switching its facility from i' to i).

2. While $U \neq \emptyset$, increase the time, and simultaneously, for every client $j \in U$, increase the parameter v_j at the same rate, until one of the following events occurs (if two events occur at the same time, we process them in an arbitrary order).

   (a) For some unopened facility i, the total offers that it receives from clients is equal to the cost of opening i. In this case, we open facility i, and for every client j (connected or unconnected) which has a non-zero offer to i, we connect j to i. The amount that j had offered to i is now called the contribution of j toward i, and j is no longer allowed to decrease this contribution.

   (b) For some unconnected client j, and some open facility i, $v_j = c_{ij}$. In this case, connect client j to facility i and remove j from U.

Since this facility location algorithm is a LMP 2-approximation FL algorithm, we can obtain a lower envelope for the k-median problem by the procedure described in Section 2.5.2. Here we run the LMP FL algorithm with different uniform facility opening costs in order to find a solution that opens exactly k facilities for each value of k. In this process, we do a binary search on the facility opening cost till a desired solution with given cardinality is reached or the tolerance for the binary search is reached. We save the solutions of different cardinalities during the binary search procedure as each one of these solutions is a 2-approximate solution for the k-median problem for that cardinality. At the end of this procedure we have a bounded envelope where the k-median solutions at the break points of the bounded envelope are 2-approximate solutions for that particular cardinality.

Solving Linear Program using CPLEX

We solve the linear programming relaxation $(k - P)$ of the standard k-median program 4.1 using the CPLEX solver.

To speed up the running time of the linear program solver, we tried to give the optimal solution of $(k-1)^{th}$ run as an initial starting solution to the iteration of cardinality k for all values of k. But there was no significant improvement of the running times of the linear programs on average.

Solving Integer Program using CPLEX

We solve the integer program $(k - IP)$ optimally the CPLEX solver. The CPLEX solver provides a way to give a good initial guess to the solver so that it can prune many low quality solutions. We give the optimal integer solution with $k - 1$ facilities as an initial guess for the CPLEX integer program iteration with cardinality k. As the optimal solution for the k-median problem for a smaller value of cardinality is a feasible solution for the k-median problem with larger cardinality, the initial guess is feasible. Even though this makes the solver find the optimal integral solution faster in some cases, it does not work in all cases and on average the improvement in running time is not significant.

$$
(k-IP) \qquad
\begin{array}{lll}
\text{Min} & \displaystyle\sum_{i\in F, j\in C} c_{ij}x_{ij} & \\
\text{subject to:} & & \\
 & \displaystyle\sum_{i\in F} x_{ij} = 1 & \forall j \in C \\
 & x_{ij} \leq y_i & \forall i \in F, \forall j \in C \\
 & \displaystyle\sum_{i\in F} y_i \leq k & \\
 & x_{ij} \in \{0,1\} & \forall i \in F, \forall j \in C \\
 & y_i \in \{0,1\} & \forall i \in F.
\end{array}
$$

4.2.2 Incremental k-median

In this section we briefly explain the Mettu and Plaxton's incremental k-median algorithm and our incremental k-median algorithm.

Mettu and Plaxton's Algorithm

Mettu and Plaxton's [48] incremental k-median algorithm uses a *hierarchical greedy approach* to choose the next facility in the incremental order to be opened. The basic idea behind this approach is as follows. Rather than selecting the next point in the ordering based on a single greedy criterion, they greedily choose a region and then recursively choose smaller regions till they arrive at a single facility which then becomes the next facility to open. Thus the choice of the next facility is influenced by a sequence of greedy criteria addressing successive finer levels of granularity.

Throughout this section, let λ, α, β and γ denote real numbers satisfying the following inequalities.

$$\begin{aligned}
\lambda &\geq 1 \\
\alpha &> 1+\lambda \\
\beta &\geq \frac{\lambda(\alpha-1)}{\alpha-1-\lambda} \\
\gamma &\geq \left(\frac{\alpha^2\beta+\alpha\beta}{\alpha-1}+\alpha\right)\lambda
\end{aligned}$$

The algorithm is listed in Algorithm 9. It uses the following additional definitions.

- A ball A is a pair (x, r) where x is the center of the ball A. We let $center(A) = x$ denote the center of the ball A and $radius(A) = r$ denote the radius of the ball A.
- Here $d(x, y)$ denotes the distance between points x and y. $d(x, Y)$ denotes the minimum distance between the point x and one of the points in the set Y.
- The value of ball $A = (x, r)$ is $\sum_{y\in A}(r - d(x, y))$ where the sum is taken over all the points within the ball A. A child of a ball (x, r) is any ball $(y, \frac{r}{\alpha})$ where $d(x, y) \leq \beta r$.

- For any point x, let *isolated*$(x, \emptyset)$ denote the ball $(x, max_{y \in F} d(x, y))$.
- For any point x and and set of facilities X, let isolated(x, X) denote the ball $(x, d(x, X)/\gamma)$.
- For any non empty sequence ρ let $head(\rho)$ (resp. $tail(\rho)$) denote the first (resp. last) element of ρ.

Algorithm 9: Mettu and Plaxton's incremental k-median algorithm

```
1. Let $Z_0 = \emptyset$. For $i = 0$ to $n - 1$, execute the following steps
    – Let $\sigma_i$ denote the singleton sequence $< A >$ where $A$ is a maximum value ball
      in $\{$isolated$(x, Z_i) | x \in F \setminus Z_i\}$
    – While the ball $tail(\sigma_i)$ has more than one child, append a maximum value child
      of $tail(\sigma_i)$ to $\sigma_i$.
    – Let $Z_{i+1} = Z_i \cup \{center(tail(\sigma_i))\}$
2. The output is the collection of facility sets $Z_i$ such that $|Z_i| = i$, $0 \leq i \leq n$ and
   $Z_i \subseteq Z_{i+1}$, $0 \leq i < n$.
```

Our incremental k-median algorithm

We implement our incremental algorithm ALTINCAPPROX described Section 2.3.2 for the incremental k-median problem on these datasets. We use Arya et al.'s local search algorithm with single swaps explained in Section 4.2.1 and the LP rounding technique of Charikar et al. [19] to generate good k-median solutions for all possible k for each of these datasets. We bucket these solutions into geometrically increasing buckets and nest the costliest solutions from each of these buckets to get nested solutions. Then we interpolate these nested solution to get incremental k-median solutions for all values of k for all datasets.

We also implement our incremental algorithm BOUNDEDINCAPPROX from Section 2.5.2 using the k-median bounded envelope obtained by running the Jain et al. algorithm explained in Algorithm 8 on the datasets. We again bucket these solutions obtained from the bounded envelope procedure and nest them using the k-median augmentation routine. By interpolating these nested solutions we get solutions for the incremental k-median problem.

4.2.3 Hierarchical k-median

We test our hierarchical k-median algorithms against the previously known hierarchical k-median algorithm by Plaxton [55].

Plaxton's Algorithm

Plaxton's algorithm takes in an incremental k-median solution as input and finds a *parent* function for each facility this incremental ordering. A hierarchical k-median solution obtained from an ordering can be considered as solutions obtained by repeatedly closing the last open facility in ordering and assigning its clients to an earlier facility. This mapping is exactly captured by the parent function in the Plaxton's algorithm. A parent function for an ordering maps every facility in the order to a facility that is earlier in the ordering. The parent of a facility is the facility that its clients will get assigned to when the facility is closed.

Plaxton's parent function is assigned as follows: Given an incremental k-median solution to the problem, a parent is assigned to every facility in the reverse order of the incremental solution. The parent of a facility f is determined by the earliest facility in the ordering that is either closer to f than any other facility occurring earlier in the ordering than f or the facility satisfying the equation in Step 3 of Algorithm 10. The equation essentially finds a facility whose distance to f is no more than the average distance of f's clients to f.

Algorithm 10: Plaxton's hierarchical k-median algorithm: Parent function calculation

1. Let $0, 1, 2..., n-1$ be the ordered incremental solution.
2. Let T_i^p be the set of clients that are assigned to i directly or to the descendants of i i.e. $T_i^p = i \cup \{T_j^p | p(j) = i\}$.
3. $p(i)$ is calculated starting from $n-1$ down to 1 as follows: $p(i)$ is set to the minimum j in $0, 1, ..., i-1$ such that $d(i,j) = \min_{k \in \{0,1,...i-1\}} d(i,k)$ or $d(i,j).|T_i^p| \le c \cdot \sum_{k \in T_i^p} d(k,i)$ for some constant c.

We run the Plaxton's parent function algorithm on the incremental k-median solutions given by running the Mettu and Plaxton's algorithm (Section 4.2.2) and ALTINCAPPROX algorithm (Section 4.2.2) using Arya et al.'s local search solutions on the datasets. We use $c = 3$ for the Plaxton's parent function calculation.

Our hierarchical k-median algorithm

We run our generic algorithm ALTINCAPPROX described Section 2.3.2 for the hierarchical k-median problem on the datasets using different k-median algorithms as black box. We use Arya et al.'s local search algorithm and Charikar et al.'s LP rounding algorithm to generate good k-median solutions. We also implement our incremental algorithm BOUNDEDINCAPPROX from Section 2.5.2 using the k-median bounded envelope obtained by running Jain et al. algorithm on the datasets. We use the hierarchical k-median augmentation routine to nest k-median solutions hierarchically after bucketing the k-median solutions.

4.3 Datasets

In our experiments we use these following sets of datasets for the comparison of k-median, incremental k-median and the hierarchical k-median algorithms.

1. *OR Library:* These 40 datasets of the uncapacitated k-median problems are part of the OR Library [8] which is a collection of test datasets for a variety of OR problems created by J.E.Beasley. These 40 test problems are named $pmed1$, $pmed2$, ..., $pmed40$ and their sizes range from $n = 100$ to 900. As noted in [8], we apply Floyd's algorithm on the adjacency cost matrix in order to obtain the complete cost matrix.
2. *Galvão:* This set of instances (*Galvão100* and *Galvão150*) is obtained from the work of Galvão and ReVelle [29]. Even though the sizes of these datasets are small ($n = 100$ and $n = 150$), the integrality gaps for some values of k (number of medians) are larger than traditional datasets.
3. *Alberta:* This dataset is generated from a 316-node network using all population centers in Alberta (see Alp, Erkut and Drezner [2]) where the distances are computed using the shortest path metric on the actual road network of Alberta.

We assume that the demand at each of the clients in the datasets is 1.

4.4 Experimental Results

4.4.1 The k-median problem

In this section we compare the performances in terms of running times and quality of solutions of five different algorithms on the datasets described: CPLEX solver for the k-median linear program, CPLEX solver for k-median integer program, Arya et al.'s single swap local search algorithm, Charikar et al.'s LP rounding algorithm and then the bounded envelope of Jain et al.'s greedy algorithm. All experiments were done on machines with Intel Core 2 2.40GHz processor with 2 gigabytes of physical memory. The linear programs and integer programs on the data sets are solved using CPLEX Version $10.1.0$. The Arya et al.'s single swap local search algorithm and Jain et al. algorithm are solved using MATLAB version 7.0. The tolerance for the bounded envelope that we use for the termination of binary search is 0.01 (see Section 2.5.2 for the bounded envelope procedure).

Tables 4.1 and 4.2 show the average and maximum ratios of the costs of the integer optimum solution (IP OPT), Arya et al.'s local optimum solutions (Local) and Charikar et al.'s LP rounding solutions (LPR) to the linear program optimum (LP OPT) over all values of k for each of the datasets. Even though the Arya et al.'s algorithm's performance guarantee is 5, in practice the local search algorithm performs much better than that. The local optimums are within 1% from the linear program optimum on average. Charikar's et al.'s LP rounding

Table 4.1: Performance of k-median algorithms

		IP OPT/LP OPT		Local /LP OPT		LPR/LP OPT	
Dataset	n	Mean	Max	Mean	Max	Mean	Max
pmed1	100	1.0001	1.01	1.0098	1.1143	1.0007	1.0532
pmed2	100	1.0001	1.0017	1.0097	1.0794	1.0035	1.1024
pmed3	100	1.0001	1.005	1.0086	1.1111	1.0029	1.0885
pmed4	100	1	1.0015	1.0022	1.0157	1.0014	1.0391
pmed5	100	1	1.002	1.0042	1.0177	1.0026	1.1311
pmed6	200	1.0004	1.0203	1.0028	1.0303	1.004	1.181
pmed7	200	1	1.0021	1.006	1.0714	1.0011	1.066
pmed8	200	1.0001	1.0084	1.0046	1.0209	1.004	1.1792
pmed9	200	1.0002	1.0071	1.0046	1.0238	1.005	1.1516
pmed10	200	1.0001	1.0097	1.0046	1.027	1.0008	1.0688
pmed11	300	1.0001	1.0039	1.0062	1.04	1.0025	1.1555
pmed12	300	1.0001	1.0149	1.0044	1.0196	1.0026	1.1285
pmed13	300	1.0001	1.0038	1.0085	1.0373	1.0026	1.0805
pmed14	300	1.0003	1.0092	1.0078	1.0253	1.0054	1.1748
pmed15	300	1	1.0024	1.007	1.0476	1.0011	1.0472
pmed16	400	1.0001	1.0087	1.0071	1.0206	1.0025	1.1701
pmed17	400	1.0001	1.0091	1.0059	1.018	1.0024	1.0977
pmed18	400	1.0002	1.0075	1.0062	1.0213	1.0037	1.1504
pmed19	400	1.0002	1.0111	1.004	1.0145	1.004	1.1506
pmed20	400	1.0001	1.0121	1.0073	1.0299	1.0022	1.1071
pmed21	500	1.0001	1.0031	1.0073	1.0227	1.004	1.1206
pmed22	500	1.0003	1.0104	1.0101	1.0789	1.0057	1.222
pmed23	500	1.0001	1.0081	1.008	1.0417	1.0044	1.1589
pmed24	500	1.0001	1.0073	1.0052	1.0238	1.0029	1.1359
pmed25	500	1.0001	1.0035	1.0062	1.0218	1.0024	1.1034

Table 4.2: (Continued)

		IP OPT/LP OPT		Local /LP OPT		LPR/LP OPT	
Dataset	n	Mean	Max	Mean	Max	Mean	Max
pmed26	600	1.0002	1.0074	1.0086	1.0448	1.005	1.1093
pmed27	600	1.0002	1.0054	1.0073	1.0435	1.0037	1.1133
pmed28	600	*	*	1.0084	1.038	1.0046	1.1578
pmed29	600	*	*	1.0079	1.0311	1.0051	1.188
pmed30	600	*	*	1.0068	1.0189	1.0036	1.1529
pmed31	700	*	*	1.0085	1.0306	1.004	1.1141
pmed32	700	*	*	1.007	1.0239	1.0048	1.2603
pmed33	700	*	*	1.0122	1.0625	1.0077	1.1875
pmed34	700	*	*	1.006	1.0205	1.0061	1.2068
pmed35	800	*	*	1.0079	1.0352	1.0053	1.2359
pmed36	800	*	*	1.0077	1.027	1.0061	1.1944
pmed37	800	*	*	1.0089	1.0323	1.0059	1.3148
pmed38	900	*	*	1.0081	1.0392	1.0042	1.2426
pmed39	900	*	*	1.0098	1.0449	1.0041	1.2423
pmed40	900	*	*	1.009	1.0543	1.0053	1.2123
Galvão100	100	1.0029	1.045	1.0041	1.0459	1.0352	1.2615
Galvão150	150	1.0048	1.0512	1.0096	1.0642	1.0479	1.2698
Alberta	316	1.0002	1.002	1.0132	1.0299	1.0026	1.0866

algorithm performs even better as most of the LP solutions are already integral or very close to being integral except for some small values of k. The $*$ in the tables represent unavailable data as the corresponding algorithmic procedure timed out.

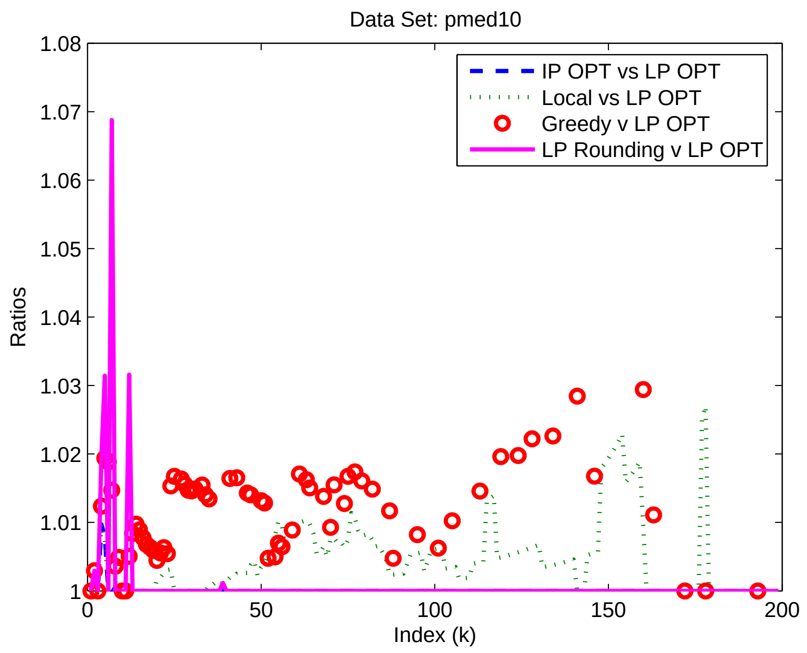

Figure 4.1: Quality of solutions of k-median algorithms (dataset $pmed10$)

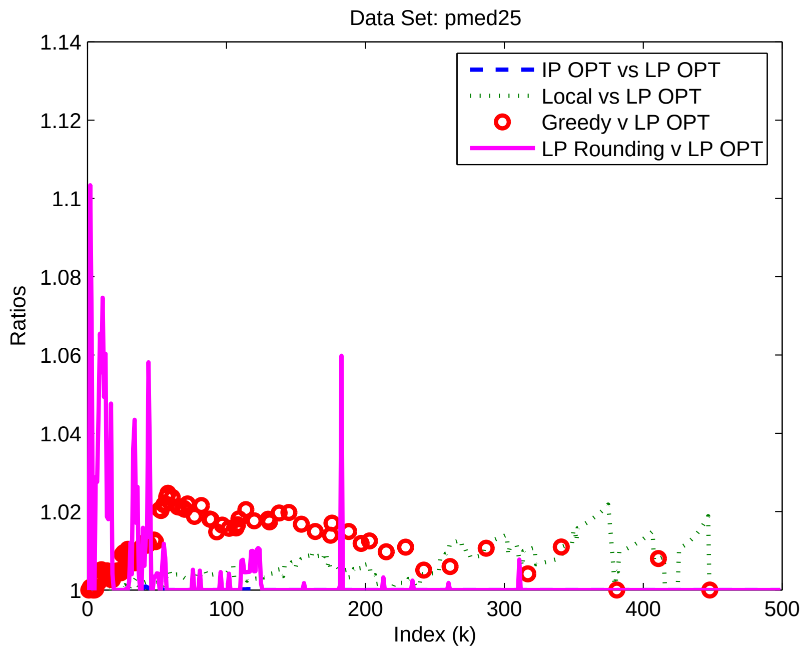

Figure 4.2: Quality of solutions of k-median algorithms (dataset $pmed25$)

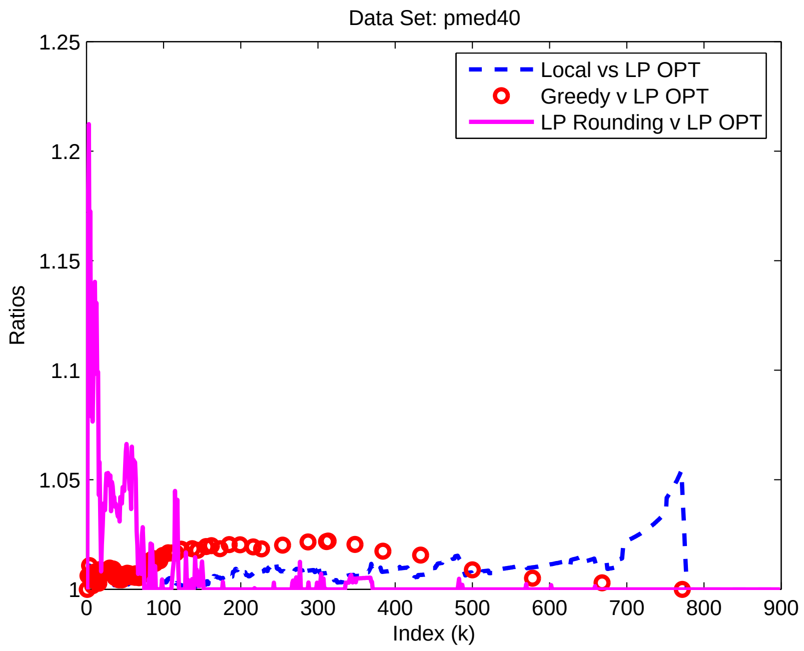

Figure 4.3: Quality of solutions of k-median algorithms (dataset $pmed40$)

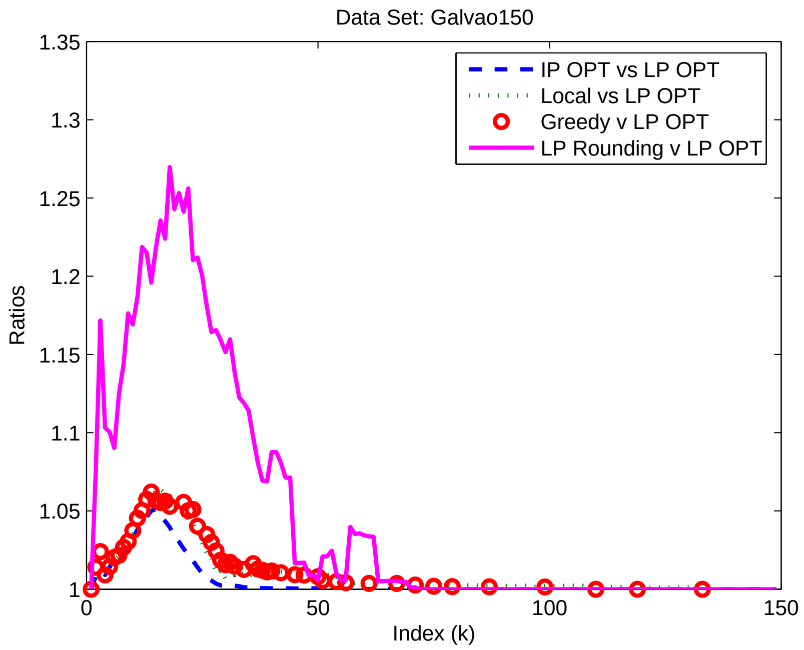

Figure 4.4: Quality of solutions of k-median algorithms (dataset *Galvão150*)

Table 4.3: Running times of k-median algorithms

		Time in seconds to finish for all n				
Dataset	n	LP	IP	Local	Greedy	LPR
pmed1	100	11.656	42.734	35.141	233.19	11.827
pmed2	100	11.547	42.609	34.734	234.92	11.735
pmed3	100	12.188	43.938	34.703	205.34	12.36
pmed4	100	11.922	39.844	34.688	197.39	12.109
pmed5	100	11.484	39.078	34.656	170.03	11.641
pmed6	200	73.203	817.52	315.94	937.38	74.374
pmed7	200	59.5	487.05	318.66	852.48	60.375
pmed8	200	64.234	603.52	317.33	979.06	65.172
pmed9	200	71.859	695.3	316.41	918.13	72.843
pmed10	200	66.859	531.39	316.89	818.47	67.75
pmed11	300	222.25	4133.9	1301.1	1927.5	225.72
pmed12	300	251.34	4158.3	1295.7	2326.8	254.47
pmed13	300	223.16	3368.8	1311.9	2397.8	226.27
pmed14	300	316.45	4979.5	1306.3	2072.1	320.31
pmed15	300	204.39	2991.9	1302.8	2001.9	207.63
pmed16	400	633.36	11117	3609.6	4033.8	641.33
pmed17	400	633.98	11853	3619	4052.7	641.27
pmed18	400	770.3	14411	3627.4	4295.1	778.7
pmed19	400	773.53	12823	3600.5	4008.1	781.13
pmed20	400	669.33	11851	3620	4186.6	677.13
pmed21	500	1427.1	34725	8122.2	5783.1	1442.7
pmed22	500	2021.1	46425	8137.7	6508.6	2037.7
pmed23	500	1494.8	34739	8128.2	6188.4	1510.1
pmed24	500	1777.1	28528	8215.2	6071.9	1791.8
pmed25	500	1223.1	29149	8372.3	6419.6	1237.8

Table 4.4: (Continued)

		Time in seconds to finish for all n				
Dataset	n	LP	IP	Local	Greedy	LPR
pmed26	600	4191.2	*	19636	10777	4219
pmed27	600	3732.3	*	16712	9733.8	3757.2
pmed28	600	3672.3	*	19768	10017	3698
pmed29	600	4020.8	*	20380	9166.4	4046.2
pmed30	600	4141.1	*	16890	9997.3	4166
pmed31	700	6859.7	*	30450	14909	6899.5
pmed32	700	7443	*	30512	15803	7481.7
pmed33	700	7786.6	*	30353	13893	7834.6
pmed34	700	*	*	30605	15411	*
pmed35	800	*	*	51954	33600	*
pmed36	800	*	*	52061	34322	*
pmed37	800	*	*	52054	33445	*
pmed38	900	*	*	*	43930	*
pmed39	900	*	*	*	49183	*
pmed40	900	*	*	*	50414	*
Galvão100	100	12.641	1208.2	35.219	90.797	13.25
Galvão150	150	40.453	40839	129.7	256.13	41.875
Alberta	316	306.59	5248.5	1642.8	3791.1	311.11

Figures 4.1, 4.2, 4.3 and 4.4, show how the costs of the k-median solutions from the integer optimum, Arya et al.'s local search algorithm, Charikar et al.'s LP rounding algorithm and the Jain et al.'s greedy algorithm compare to the linear program for different values of k for four sample datasets $pmed10$[1], $pmed25$[1], $pmed40$ and *Galvão150*. Note that the Jain et al.'s greedy LMP FL algorithm gives only a bounded envelope and does not give k-median solutions for all values

[1]Most of "IP OPT v LP OPT" and "LP rounding v LP OPT" plot coincides with x-axis

of k. Here we can see that the LP rounding algorithm and the local search algorithm performs better than the Jain et al.'s algorithm. Also note that the IP optimal solutions cost almost the same as the LP optimum solutions since most of the LP optimum solutions are integral but performs worse than local search for some small values of k.

Tables 4.3 and 4.4 give the times in seconds of the runs of each of the above mentioned algorithms on each of the data sets for all values of k. For the LP and IP columns, each time entry denotes the sum of the times taken for the $CPLEX$ solves over all values of k for that particular dataset. Note that the LP solver runs faster than the local search and greedy algorithm for all datasets. Also note that the IP solver takes a lot more time to solve all the instances of k for bigger datasets.

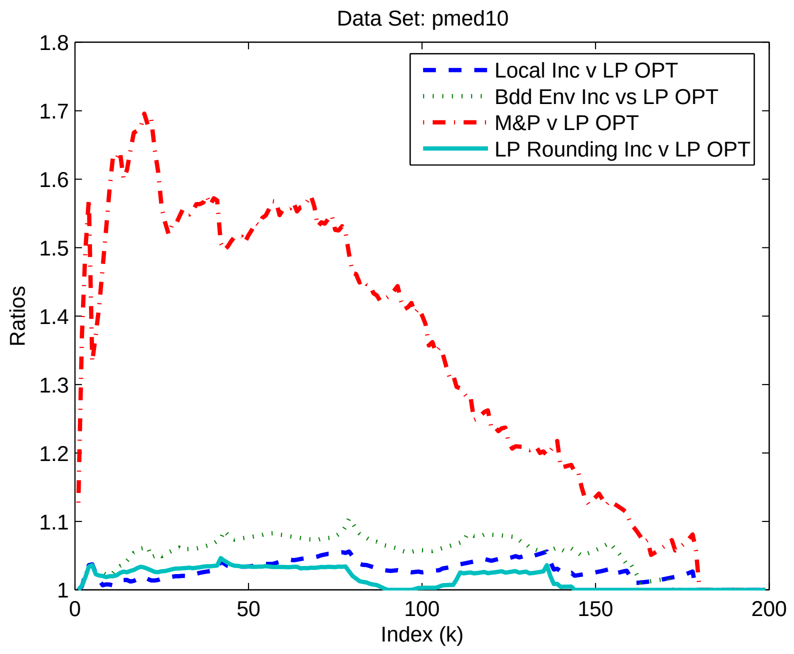

Figure 4.5: Quality of solutions of incremental k-median algorithms (dataset $pmed10$)

Table 4.5: Performance of incremental k-median algorithms

		LInc/LP OPT		GInc/LP OPT		MPInc/LP OPT		LPRInc/LP OPT	
Dataset	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max
pmed1	100	1.0397	1.1143	1.0808	1.1298	1.2388	1.5449	1.0548	1.1018
pmed2	100	1.0253	1.0587	1.0356	1.0869	1.2596	1.4596	1.0069	1.0332
pmed3	100	1.0297	1.1379	1.0583	1.1339	1.1989	1.3753	1.0248	1.0726
pmed4	100	1.0044	1.0333	1.0419	1.0978	1.2093	1.6107	1.0099	1.0333
pmed5	100	1.0078	1.03	1.0619	1.1593	1.1812	1.4791	1.0127	1.0938
pmed6	200	1.0115	1.0603	1.0273	1.0603	1.2287	1.4456	1.0103	1.0603
pmed7	200	1.0211	1.0714	1.0526	1.0985	1.2742	1.4944	1.0152	1.0482
pmed8	200	1.0142	1.0519	1.0241	1.0554	1.2502	1.5199	1.0089	1.0398
pmed9	200	1.0171	1.0476	1.0308	1.0479	1.3135	1.6001	1.0073	1.0392
pmed10	200	1.0276	1.0562	1.0521	1.1043	1.3347	1.6957	1.0168	1.0467
pmed11	300	1.0205	1.04	1.0522	1.0998	1.3153	1.6443	1.0143	1.035
pmed12	300	1.0163	1.0408	1.0521	1.1121	1.3018	1.5773	1.0133	1.0432
pmed13	300	1.0236	1.0554	1.0402	1.0729	1.3329	1.6403	1.0213	1.0612
pmed14	300	1.0216	1.0458	1.0572	1.1061	1.2594	1.6084	1.0163	1.1177
pmed15	300	1.0214	1.0544	1.0404	1.0721	1.3826	1.7325	1.018	1.0306
pmed16	400	1.0255	1.0469	1.0399	1.0755	1.3904	1.7797	1.0228	1.1024
pmed17	400	1.015	1.0488	1.0574	1.0891	1.3286	1.6403	1.0267	1.0575
pmed18	400	1.0227	1.0597	1.0486	1.0864	1.3195	1.5553	1.0225	1.1216
pmed19	400	1.0102	1.0462	1.0384	1.0585	1.3051	1.5939	1.0222	1.0727
pmed20	400	1.0273	1.0489	1.0385	1.0757	1.2429	1.4529	1.0256	1.0691
pmed21	500	1.023	1.0439	1.0425	1.0843	1.2958	1.6379	1.0235	1.0587
pmed22	500	1.0254	1.0789	1.0533	1.0973	1.3233	1.6869	1.0165	1.0573
pmed23	500	1.0249	1.0417	1.0354	1.0714	1.3228	1.5863	1.0236	1.0482
pmed24	500	1.0165	1.0441	1.0356	1.0687	1.3197	1.57	1.0193	1.0595
pmed25	500	1.0204	1.0395	1.0399	1.071	1.3522	1.7349	1.0215	1.0478

Table 4.6: (Continued)

		LInc/LP OPT		GInc/LP OPT		MPInc/LP OPT		LPRInc/LP OPT	
Dataset	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max
pmed26	600	1.0224	1.0496	1.0479	1.087	1.3655	1.6944	1.0197	1.0627
pmed27	600	1.0207	1.0475	1.0378	1.0781	1.3266	1.6875	1.0198	1.0533
pmed28	600	1.0302	1.0549	1.0398	1.0703	1.3821	1.8071	1.021	1.07
pmed29	600	1.0226	1.0365	1.0401	1.0774	1.3452	1.7871	1.0209	1.1158
pmed30	600	1.0179	1.0466	1.0362	1.0722	1.3233	1.8021	1.0192	1.0639
pmed31	700	1.0228	1.0441	1.0383	1.0817	1.3608	1.7558	1.0242	1.0588
pmed32	700	1.0178	1.0424	1.0362	1.0704	1.373	1.769	1.0154	1.0771
pmed33	700	1.0365	1.0627	1.049	1.0826	1.3381	1.7314	1.0237	1.0691
pmed34	700	1.0202	1.049	1.0374	1.07	1.3768	1.7976	1.0172	1.0836
pmed35	800	1.0231	1.1336	1.0367	1.0708	1.4248	1.9212	1.0163	1.1336
pmed36	800	1.0242	1.0411	1.035	1.0645	1.3849	1.8291	1.0229	1.0665
pmed37	800	1.0204	1.041	1.0356	1.0709	1.3405	1.7722	1.0189	1.0853
pmed38	900	1.0208	1.0392	1.0394	1.0688	1.3901	1.7921	1.0237	1.0656
pmed39	900	1.0245	1.0449	1.0451	1.0842	1.384	1.828	1.0172	1.0449
pmed40	900	1.0194	1.0543	1.035	1.0689	1.3397	1.7058	1.0208	1.161
Galvão100	100	1.0209	1.1601	1.024	1.1601	1.0711	1.3496	1.036	1.2205
Galvão150	150	1.0283	1.1655	1.0349	1.1925	1.0908	1.388	1.0285	1.2168
Albert a	316	1.0409	1.1967	1.0499	1.1432	1.157	1.4322	1.027	1.1223

4.4.2 Incremental k-median

In this section we compare the performances of four different incremental k-median algorithms on the selected datasets: Mettu and Plaxton's incremental k-median algorithm (MPInc), our ALTINCAPPROX algorithm with solutions from the Arya et al.'s single swap local search algorithm (LInc) and Charikar et al's LP rounding (LPR) and our BOUNDEDINCAPPROX algorithm with the bounded

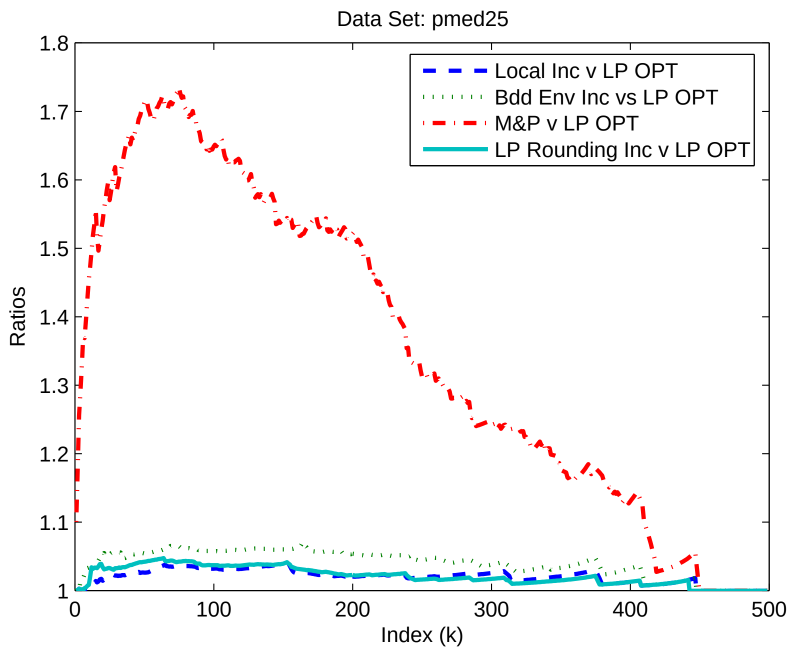

Figure 4.6: Quality of solutions of incremental k-median algorithms (dataset $pmed25$)

envelope obtained from the Jain et al.'s greedy LMP FL algorithm (GInc). We pick $\lambda = 1, \alpha = 3, \beta = 2, \gamma = 15$ for the Mettu and Plaxton's algorithm simulations so that they satisfy the equations in Section 4.2.2.

The third and fourth column of Tables 4.5 and 4.6 shows the average and maximum ratios of the costs of the incremental k-median solution obtained from the ALTINCAPPROX algorithm using Arya et al.'s local search k-median solutions (LInc) to the linear program optimum (LP OPT) for each of the datasets. The fifth and sixth columns give the corresponding average and maximum value ratios for the incremental k-median solution costs of the BOUNDEDINCAPPROX using bounded envelope obtained by running Jain et al's algorithm in Section 4.2.1. The next two columns give the corresponding average and maximum ratios for the Mettu and Plaxton's incremental k-median algorithm

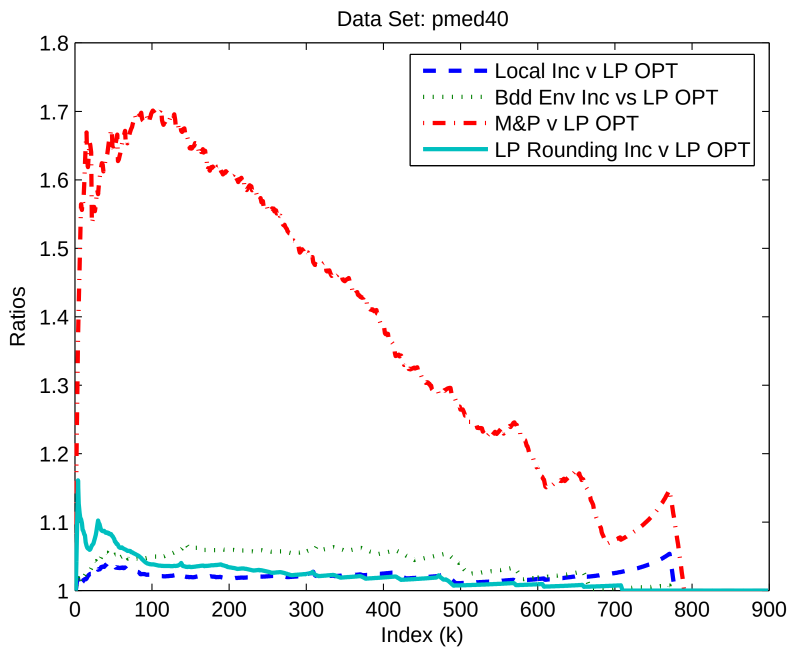

Figure 4.7: Quality of solutions of incremental k-median algorithms (dataset $pmed40$)

and the last two columns give the ratios for Charikar et al's LP rounding algorithm. From the tables we infer that our algorithms perform much better than the Mettu and Plaxton's algorithm on the datasets. This inference is reinforced by the Figures 4.5, 4.6, 4.7 and 4.8 which show that the ratios of the costs of solutions obtained from our incremental algorithms to the LP optimum are always better than the corresponding ratios of Mettu and Plaxton's algorithm for a sample of 4 datasets ($pmed10$, $pmed25$, $pmed40$ and *Galvão*).

Tables 4.7 and 4.8 give the times in seconds of the runs of each of the above mentioned algorithms on each of the data sets for all values of k. Note that the Mettu and Plaxton's algorithm runs much faster than the our algorithms which uses a k-median algorithm or a bounded envelope algorithm as a black-box which make it very slow. However the quality of the incremental solutions

Table 4.7: Running times of incremental k-median algorithms

		Running time in secs			
Dataset	Size	IncL	IncG	MP	IncLPR
pmed1	100	35.266	233.31	0.593	11.999
pmed2	100	34.859	235.06	0.594	11.907
pmed3	100	34.844	205.47	0.578	12.532
pmed4	100	34.829	197.52	0.61	12.281
pmed5	100	34.766	170.17	0.578	11.797
pmed6	200	316.83	938.33	2.375	75.327
pmed7	200	319.55	853.36	2.359	61.313
pmed8	200	318.22	979.92	2.36	66.063
pmed9	200	317.28	919	2.343	73.796
pmed10	200	317.8	819.36	2.375	68.672
pmed11	300	1304.7	1930.9	5.672	229.03
pmed12	300	1299	2330.2	5.938	257.84
pmed13	300	1315.4	2401.1	5.766	229.55
pmed14	300	1309.6	2075.4	5.625	323.61
pmed15	300	1306.1	2005.2	5.797	210.95
pmed16	400	3617.8	4042	11.218	649.55
pmed17	400	3627.4	4061	10.844	649.7
pmed18	400	3636.1	4303.4	11.844	786.97
pmed19	400	3608.9	4016.3	12.125	789.36
pmed20	400	3628.8	4194.8	11.031	685.36
pmed21	500	8139.7	5800.6	18.438	1460.2
pmed22	500	8155.5	6526.2	18.547	2055.4
pmed23	500	8146.3	6206.6	18.485	1527.6
pmed24	500	8233.5	6089.8	18.578	1809.4
pmed25	500	8390.4	6437.8	18.813	1255.2

Table 4.8: (Continued)

		Running time in secs			
Dataset	Size	IncL	IncG	MP	IncLPR
pmed26	600	19667	10807	30	4249.8
pmed27	600	16743	9764.5	29.969	3788.9
pmed28	600	19799	10048	29.86	3730
pmed29	600	20411	9197.1	30.157	4077
pmed30	600	16921	10029	30.297	4196.9
pmed31	700	30499	14958	44.891	6949.1
pmed32	700	30561	15854	43.657	7532.4
pmed33	700	30404	13943	45.422	7884
pmed34	700	30655	15461	43.391	*
pmed35	800	52037	33683	61.454	*
pmed36	800	52141	34402	62.376	*
pmed37	800	52134	33525	61.735	*
pmed38	900	*	44037	82.595	*
pmed39	900	*	49290	78.954	*
pmed40	900	*	50522	81.673	*
Galvão100	100	35.36	90.937	0.563	13.422
Galvão150	150	130.05	256.45	1.281	42.265
Alberta	316	1646.6	3794.9	6.375	315.03

obtained from our algorithm is much better than the Mettu and Plaxton's algorithm which compensates for the extra time it takes.

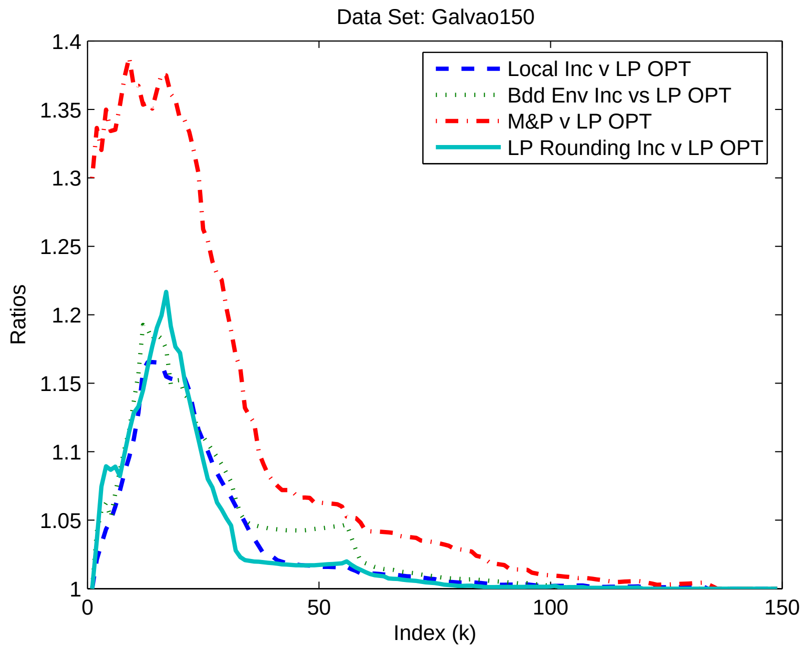

Figure 4.8: Quality of solutions of incremental k-median algorithms (dataset *Galvão150*)

4.4.3 Hierarchical k-median

In this section we compare the performances of Plaxton's hierarchical k-median algorithm against our ALTINCAPPROX hierarchical k-median algorithm on the datasets. Note that Plaxton's algorithm takes in any incremental k-median solution as input and outputs a parent function which defines the hierarchical solution. We give the incremental k-median solutions from the algorithm runs of ALTINCAPPROX and the Mettu and Plaxton's algorithms as input to the Plaxton's hierarchical algorithm (PHLI and PHMP) and compare them against our hierarchical k-median algorithms' solutions (HL, HG and LPRH) for different datasets.

Figures 4.9, 4.10, 4.11 and 4.12 show how the costs of the hierarchical k-

Table 4.9: Performance of hierarchical k-median algorithms

		HL/LP		HG/LP		PHLI/LP		PHMP/LP		LPRH/LP	
Dataset	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
pmed1	100	1.05	1.11	1.1	1.17	1.07	1.3	1.35	1.83	1.07	1.14
pmed2	100	1.04	1.13	1.05	1.13	1.04	1.17	1.31	1.56	1.02	1.13
pmed3	100	1.04	1.16	1.07	1.14	1.06	1.24	1.31	1.57	1.04	1.18
pmed4	100	1.02	1.1	1.06	1.14	1.04	1.24	1.3	1.85	1.02	1.1
pmed5	100	1.02	1.12	1.07	1.16	1.01	1.11	1.27	1.78	1.02	1.18
pmed6	200	1.03	1.17	1.04	1.14	1.04	1.23	1.34	1.74	1.03	1.16
pmed7	200	1.04	1.14	1.08	1.17	1.05	1.23	1.39	1.78	1.04	1.16
pmed8	200	1.03	1.12	1.04	1.15	1.06	1.29	1.33	1.71	1.03	1.11
pmed9	200	1.04	1.16	1.05	1.16	1.05	1.17	1.45	2.04	1.03	1.14
pmed10	200	1.04	1.14	1.07	1.13	1.05	1.17	1.45	1.99	1.03	1.15
pmed11	300	1.04	1.11	1.07	1.14	1.06	1.23	1.45	1.88	1.03	1.13
pmed12	300	1.04	1.17	1.07	1.18	1.06	1.31	1.46	1.95	1.03	1.15
pmed13	300	1.04	1.17	1.06	1.14	1.06	1.27	1.48	2.03	1.04	1.17
pmed14	300	1.04	1.18	1.08	1.18	1.06	1.39	1.42	1.99	1.04	1.25
pmed15	300	1.04	1.15	1.06	1.19	1.05	1.19	1.58	2.22	1.04	1.16
pmed16	400	1.05	1.18	1.06	1.2	1.07	1.3	1.62	2.4	1.04	1.19
pmed17	400	1.04	1.23	1.08	1.21	1.06	1.28	1.48	1.96	1.05	1.19
pmed18	400	1.05	1.21	1.07	1.18	1.08	1.35	1.49	1.96	1.05	1.22
pmed19	400	1.03	1.17	1.06	1.19	1.05	1.29	1.44	1.93	1.04	1.2
pmed20	400	1.05	1.17	1.07	1.2	1.07	1.3	1.35	1.74	1.05	1.17
pmed21	500	1.05	1.16	1.06	1.16	1.06	1.26	1.46	2	1.05	1.16
pmed22	500	1.05	1.2	1.08	1.2	1.07	1.35	1.57	2.32	1.04	1.26
pmed23	500	1.05	1.17	1.06	1.17	1.06	1.27	1.49	2.03	1.05	1.2
pmed24	500	1.04	1.18	1.06	1.19	1.06	1.34	1.48	1.96	1.04	1.2
pmed25	500	1.04	1.14	1.06	1.18	1.06	1.26	1.54	2.11	1.04	1.15

Table 4.10: (Continued)

		HL/LP		HG/LP		PHLI/LP		PHMP/LP		LPRH/LP	
Dataset	n	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
pmed26	600	1.05	1.18	1.07	1.18	1.07	1.3	1.54	2.2	1.04	1.18
pmed27	600	1.05	1.18	1.06	1.18	1.06	1.33	1.49	2.08	1.05	1.18
pmed28	600	1.05	1.19	1.06	1.19	1.08	1.32	1.57	2.34	1.05	1.22
pmed29	600	1.05	1.19	1.06	1.19	1.06	1.35	1.53	2.28	1.05	1.24
pmed30	600	1.04	1.19	1.06	1.18	1.06	1.32	1.54	2.3	1.04	1.25
pmed31	700	1.04	1.19	1.06	1.19	1.06	1.37	1.56	2.31	1.04	1.2
pmed32	700	1.05	1.22	1.07	1.24	1.06	1.33	1.54	2.27	1.04	1.23
pmed33	700	1.06	1.19	1.07	1.22	1.08	1.39	1.54	2.36	1.05	1.22
pmed34	700	1.05	1.19	1.06	1.18	1.07	1.31	1.58	2.23	1.04	1.29
pmed35	800	1.05	1.29	1.06	1.17	1.07	1.39	1.64	2.52	1.04	1.31
pmed36	800	1.05	1.24	1.06	1.25	1.08	1.38	1.6	2.38	1.05	1.22
pmed37	800	1.05	1.24	1.06	1.21	1.07	1.42	1.52	2.26	1.05	1.24
pmed38	900	1.05	1.19	1.06	1.2	1.06	1.36	1.62	2.44	1.05	1.21
pmed39	900	1.05	1.21	1.07	1.23	1.07	1.35	1.55	2.27	1.05	1.22
pmed40	900	1.05	1.22	1.06	1.23	1.07	1.4	1.53	2.22	1.05	1.3
Galvão100	100	1.05	1.27	1.05	1.36	1.07	1.37	1.24	1.81	1.09	1.43
Galvão150	150	1.08	1.34	1.08	1.37	1.1	1.44	1.28	1.84	1.09	1.35
Alberta	316	1.06	1.27	1.07	1.25	1.08	1.28	1.27	1.72	1.05	1.22

median solutions for different algorithms compare against the optimal linear program solutions for different values of k for four sample datasets $pmed10$, $pmed25$, $pmed40$ and *Galvão150*. The algorithms we consider are ALTINCAPPROX algorithm (using Arya et al.'s local search k-median solutions (HL) and Charikar et al.'s LP rounding solutions (LPRH)), BOUNDEDINCAPPROX algorithm (using bounded envelope from Jain et al.'s greedy algorithm) (HG), Plaxton's hierarchical k-median algorithm on the incremental solutions of ALTINcAPPROX algorithm (PHLI) and Plaxton's algorithm on Mettu and Plaxton's incremental k-median solutions (PHMP).

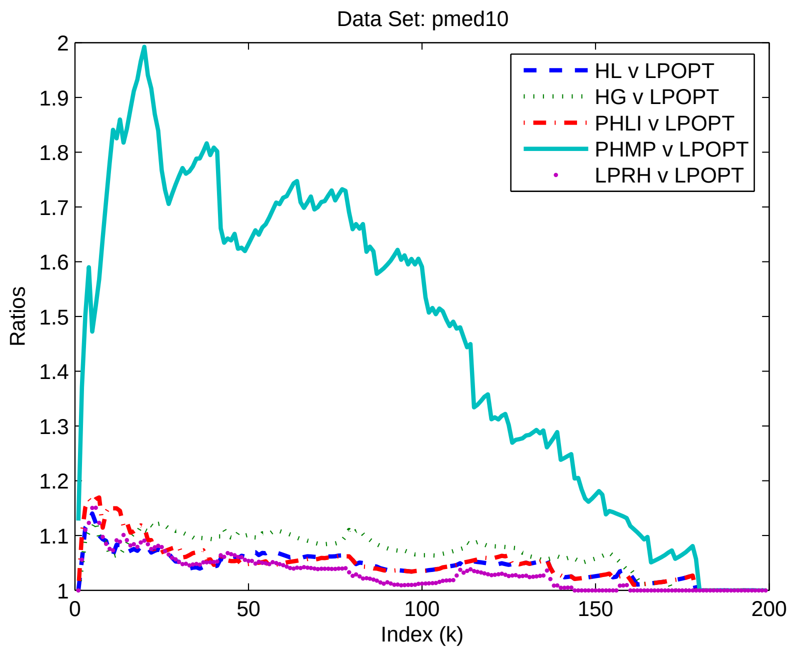

Figure 4.9: Quality of solutions of hierarchical k-median algorithms (dataset $pmed10$)

Tables 4.9 and 4.10 show the average and maximum ratios of the costs of the hierarchical k-median solutions of different hierarchical algorithms (HL, HG, PHLI, PHMP) to the cost of linear program optimum (LP OPT) for each of the datasets.

We can see clearly that the hierarchical solutions obtained by ALTINCAPPROX algorithms are better than other algorithms. Note that the ratios for the PHMP algorithm are not as good as for the other algorithms since PHMP uses the incremental k-median solutions of Mettu and Plaxton as input which are not as good as other incremental algorithms in terms of quality (See Tables 4.5 and 4.6). Our hierarchical algorithm (HL) which computes hierarchical solutions di-

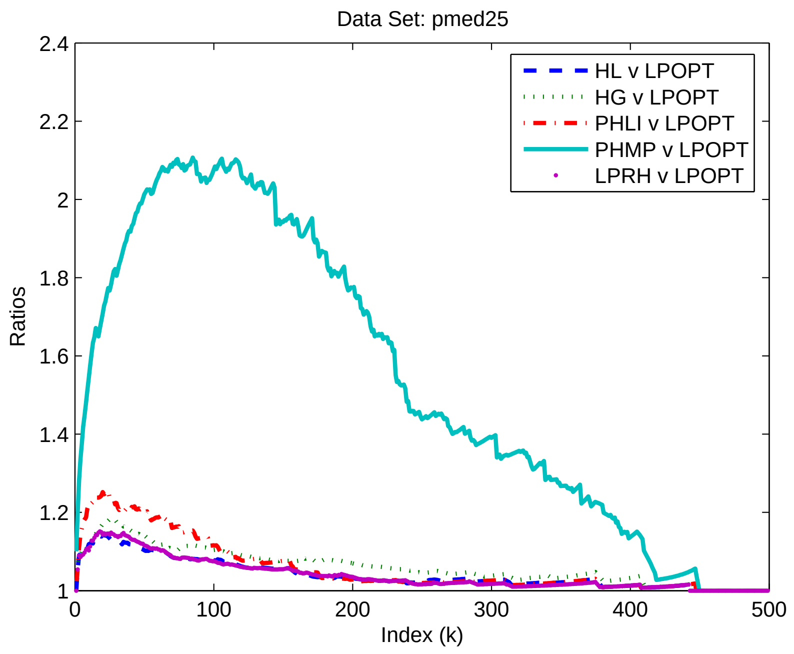

Figure 4.10: Quality of solutions of hierarchical k-median algorithms (dataset $pmed25$)

rectly from k-median solutions performs better than the Plaxton's hierarchical algorithm even when the incremental solutions from ALTINCAPPROX are given as input. We do not provide a table with the running times for different hierarchical algorithms as the major chunk of the running times is contributed by the underlying incremental k-median or k-median algorithms.

4.5 Conclusion

We simulate different k-median, incremental k-median and hierarchical k-median algorithms on different datasets and show our results here. For the k-median problem, Charikar et al.'s LP rounding algorithm performs better and faster on average than other k-median algorithms like Arya et al.'s local search algorithm.

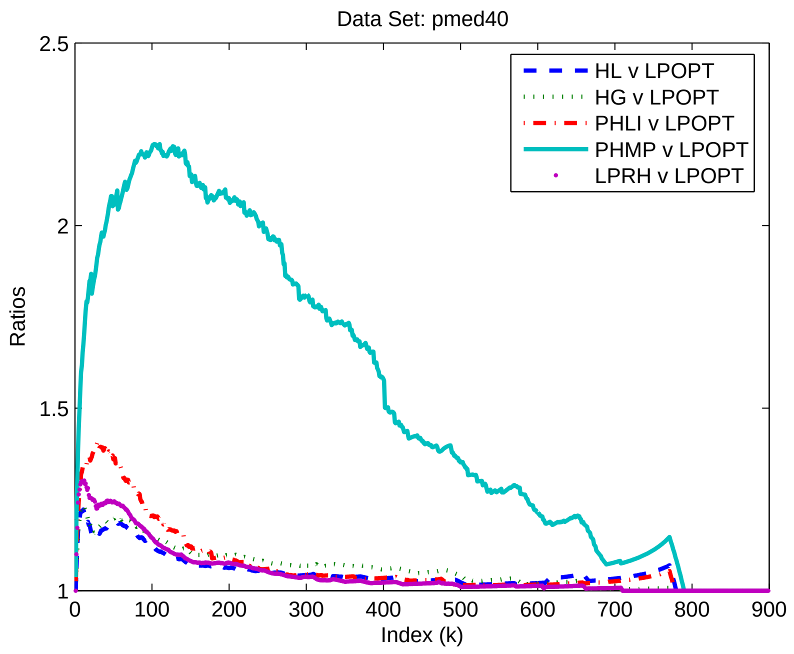

Figure 4.11: Quality of solutions of hierarchical k-median algorithms (dataset $pmed40$)

We also notice that in many real-life datasets the optimal LP solution for the k-median problems for most values of k are integers which also makes the LP rounding techniques much better in terms of the quality of the solutions.

The quality of incremental solutions, when ALTINCAPPROX algorithm is run on the k-median solutions of Arya et al's local search algorithm and Charikar et al's LP rounding algorithm, are much better than the incremental solutions of Mettu and Plaxton's algorithm. Even though the LP rounding algorithm performs poorly for some small values of k, our incremental and hierarchical algorithms skips many of these poor solutions while bucketing the solutions geometrically and this makes the corresponding incremental solutions comparable in quality to the incremental solutions obtained from Arya et al.'s local search k-median solutions.

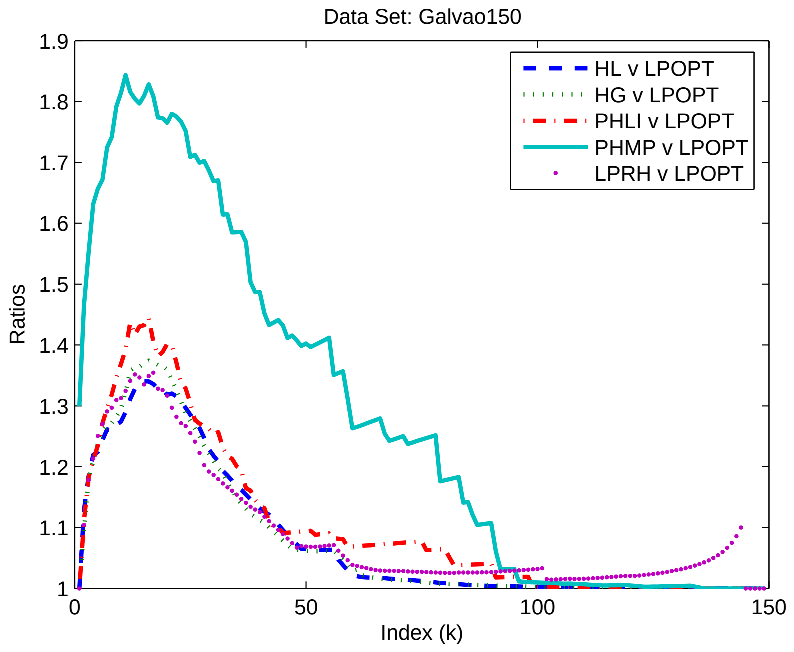

Figure 4.12: Quality of solutions of hierarchical k-median algorithms (dataset *Galvão150*)

It is easy to infer that Mettu and Plaxton's incremental k-median algorithm is a much quicker algorithm compared to other incremental k-median algorithms we implement. However one important point to note here is that we find good k-median solutions for all values of k both in Arya et al.'s local search algorithm and Charikar et al.'s LP rounding algorithm. Most of these solutions are not used at all since we use only one solution from each of the geometrically increasing buckets. Instead we can find k-median solutions for some selected values of k and nest them using the nesting algorithm to get good incremental and hierarchical solutions. This simple improvement in our implementation can improve the running times of our algorithms in Tables 4.3, 4.4, 4.7 and 4.8 tremendously and make them comparable to that of Mettu and Plaxton's algorithm.

Observing the results from Tables 4.5, 4.6, 4.9 and 4.10, we can infer that the deviations in the costs of the solutions produced by our incremental k-median and hierarchical k-median algorithms (using ALTINCAPPROX algorithm) from optimal costs is much lower compared to that of other existing algorithms. So our ALTINCAPPROX and BOUNDEDINCAPPROX algorithms give better performance guarantees in theory and in experiments while simultaneously not compromising too much on the algorithms' running times which is ideal for any practical incremental and hierarchical problem.

BIBLIOGRAPHY

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The online set cover problem. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 100–105, 2003.

[2] O. Alp, E. Erkut, and D. Drezner. An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, 122:21–42(22), 2003.

[3] B. M. Anthony and A. Gupta. Infrastructure leasing problems. In *Proceedings of Integer Programming and Combinatorial Optimization*, pages 424–438, 2007.

[4] A. Archer, A. Levin, and D. P. Williamson. A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing*, 37:1472–1498, 2008.

[5] A. Archer and D. P. Williamson. Faster approximation algorithms for the minimum latency problem. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, pages 88–96, 2003.

[6] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33:544–562, 2004.

[7] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[8] J. E. Beasley. A note on solving large p-median problems. *European Journal of Operational Research*, 21:270–273, 1985.

[9] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23:769 – 805, 1998.

[10] A. Ben-Tal and A. Nemirovski. Robust solutions to uncertain programs. *Operations Research Letters*, 25:1–13, 1999.

[11] P. Berman and C. Coulston. On-line algorithms for Steiner tree problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 344–353, 1997.

[12] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98:49–71, 2003.

[13] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 163–171, 1994.

[14] A. Borodin and R. El-Yaniv, editors. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[15] N. H. Bshouty and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *The Proceedings of the 15th Annual Symposium on the Theoretical Aspects of Computer Science*, pages 298–308, 1998.

[16] J. Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. Report PNA-E0611, CWI, Nov. 2006.

[17] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[18] M. Charikar, C. Chekuri, A. Goel, and S. Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-median. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 114–123, 1998.

[19] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the 31th Annual ACM Symposium on Theory of Computing*, pages 1–10, 1999.

[20] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651, 2001.

[21] M. Chrobak, C. Kenyon, J. Noga, and N. E. Young. Incremental medians via online bidding. *Algorithmica*, 50:455–478, 2008.

[22] M. Chrobak, C. Kenyon, and N. Young. The reverse greedy algorithm for the metric k-median problem. In *Information Processing Letters 97*, pages 68–72, 2006.

[23] F. A. Chudak and D. Shmoys. Improved approximation algorithms for uncapacitated facility location. *SIAM Journal on Computing*, 33(1):1–25, 2004.

[24] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233–235, 1979.

[25] B. Codenotti, G. De Marco, M. Leoncini, M. Montangero, and M. Santini. Approximation algorithms for a hierarchically structured bin packing problem. *Information Processing Letters*, 89:215–221, 2004.

[26] S. Dasgupta and P. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70:555–569, 2005.

[27] D. Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2007.

[28] D. Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5:141–148, 2007.

[29] R. D. Galvao and C. ReVelle. A Lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, 88(1):114–123, 1996.

[30] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation algorithms for partial covering problems. In *Proceedings of the 11th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 225–236, 2001.

[31] N. Garg. Saving an epsilon: A 2-approximation for the k-MST problem in graphs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 396–402, 2005.

[32] M. X. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82:111–124, 1998.

[33] T. González. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[34] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, pages 649–657, 1998.

[35] A. Gupta, M. Pal, R. Ravi, and A. Sinha. Boosted sampling: Approximation

algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 417–426, June 2004.

[36] J. Hartline and A. Sharp. Hierarchical flow. In *Proceedings of the 2005 International Network Optimization Conference (INOC 2005)*, pages 681–687, 2005.

[37] D. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 111–122, 1998.

[38] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[39] D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, Boston, MA, 1995.

[40] N. Immorlica, D. Karger, M. Minkoff, and V. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 691–700, 2004.

[41] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50:795–824, 2003.

[42] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48:274–296, 2001.

[43] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. Universal approximations for tsp, Steiner tree, and set cover. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 386–395, 2005.

[44] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[45] J. Lin and J. Vitter. Approximation algorithms for geometric median problems. Technical report, 1992.

[46] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithm for

metric facility location problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 229–242. Springer, 2002.

[47] J. Mestre. A primal-dual approximation algorithm for partial vertex cover: making educated guesses. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, volume 3624, pages 182–191, 2005.

[48] R. R. Mettu and C. G. Plaxton. The online median problem. *SIAM Journal on Computing*, 32:816–832, 2003.

[49] A. Meyerson. Online facility location. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.

[50] A. Meyerson. The parking permit problem. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 274–282, 2005.

[51] Y. Nikulin. Robustness in combinatorial optimization and scheduling theory: An annotated bibliography. http://www.optimization-online.org/DB_FILE/2004/11/995.pdf.

[52] Odyssey Logistics. See `http://www.odysseylogistics.com`.

[53] L. K. Platzman and J. J. Bartholdi. Spacefilling curves and the planar travelling salesman problem. *Journal of ACM*, 36(4):719–737, 1989.

[54] C. G. Plaxton. Approximation algorithms for hierarchical location problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 40–49, 2003.

[55] C. G. Plaxton. Approximation algorithms for hierarchical location problems. *Journal of Computer and System Sciences*, 72:425–443, 2006.

[56] R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108(1):97–114, 2006.

[57] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[58] P. Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64:251–254, 1997.

[59] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–205, 1985.

[60] A. Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1305–1313, 2007.

[61] C. Swamy and D. B. Shmoys. Sampling-based approximation algorithms for multi-stage stochastic optimization. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 357–366, 2005.

[62] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.